Response time minimization in distributed query optimization.

Mohan Kumar S. Bethur
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/3341

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000 ext. 3208.
NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.
Response Time Minimization in Distributed Query Optimization

by

Mohan Kumar S. Bethur

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through the School of Computer Science in Partial
Fulfillment of the Requirements for the Degree of
Master of Science at the
University of Windsor

Windsor, Ontario, Canada
1995
The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

Abstract

In Distributed Database Systems, the principal objective is to find an execution strategy which minimizes the cost. To find the best strategy, the query processing strategies which are commonly used include joins, semijoins and improvement algorithms. Here, in this thesis, a semijoin query processing strategy is used to find the best execution strategy.

In the AHY (Apers-Hevner-Yao) algorithms, the investigations only focus on reducing the amount of transmissions. They make the assumption that the cost to send the packets from any source to any destination is the same and they don't take into consideration the differences in delays in the links on the network. The objective of this thesis is to develop a heuristic which will take the network load along with the size of the data to be transmitted and compare it to the AHY algorithm GENERAL (Response time version).
Dedicated to my
father B.M. Siddaiah,
mother H.G. Shivalingamma,
sisters Shakuntala and Sunitha,
and brother Vijay Kumar
Acknowledgments

My journey towards the pursuit of my Master’s thesis has been true with the work on my thesis and writing up of this thesis report. I would like to express my gratitude to all my friends, fellow students and friends who have made this “journey” such a positive and a rewarding experience for me.

I want to express my gratitude to Dr. Richard Frost for accepting me into the Master’s program, for his guidance through my Master’s program and also for providing an excellent work environment.

I would like to express my sincere thanks and appreciation to Dr. Subir Bandyopadhyay and Dr. Joan Morrissey for their support and guidance throughout the progress of this thesis and for helping me develop into a successful Master’s candidate.

I want to thank Dr. Young Park for being my internal reader and for his valuable suggestions. I would also like to thank Prof. Phil Alexander, my external reader, for his comments on my thesis.

I want to thank Todd Bealor and Sandeep Karnat for their comments, help and cooperation.

Finally, I would like to thank my parents, my sisters and my brother for their patience, support and encouragement and helping my dream come true. Lastly, I would like to thank my uncle K.G. Basavarajappa and aunty Dakshayini Basavaraj for their support and encouragement.

Thank you all very much.
TABLE OF CONTENTS

Abstract ................................................................. iv
Acknowledgments ......................................................... vi
LIST OF FIGURES .......................................................... x
LIST OF TABLES ............................................................. xi

1 INTRODUCTION ............................................................ 1
1.1 PROBLEM TO BE INVESTIGATED ................................. 2
1.2 THE THESIS STATEMENT ............................................ 3
1.3 OBJECTIVES AND SCOPE OF THE THESIS WORK ............. 3
1.4 ORGANIZATION OF THE THESIS REPORT ...................... 3

2 BACKGROUND : REVIEW OF THE LITERATURE .................... 4
2.1 DISTRIBUTED QUERY PROCESSING AND OPTIMIZATION ......... 4

Distributed Query Processing ......................................... 6
Semijoin Query Processing Strategy ................................. 7

2.2 COST ESTIMATION TECHNIQUES ................................... 9

Cost Measures ........................................................... 9
Database Statistics ..................................................... 11

2.3 SEMIJOIN QUERY OPTIMIZATION STRATEGIES ................ 14

2.4 AHY ALGORITHM .................................................... 17

Algorithm PARALLEL .................................................. 17
Algorithm SERIAL ...................................................... 18
Algorithm GENERAL ................................................... 19
2.5 COMPUTER NETWORKS ............................................. 22
   Network Topology ............................................. 22
   The Physical Transmission of Data .............................. 23
   Architecture and Importance of Computer Networks ......... 24
   FTP and PING .................................................. 26

3 HEURISTIC FOR QUERY OPTIMIZATION ......................... 27
   3.1 COST MODEL OF THE HEURISTIC ............................... 27
   3.2 ASSUMPTIONS, DEFINITIONS AND NOTATIONS ............... 29
   3.3 DESCRIPTION OF THE HEURISTIC ............................. 31
   3.4 EXAMPLE OF THE HEURISTIC .................................. 34
      Step I ....................................................... 35
      Step II ..................................................... 36
      Step III .................................................... 37
      Step IV ..................................................... 38
      Step V ....................................................... 39

4 EXPERIMENTAL RESULTS ........................................... 40
   4.1 FTP AND PING RESULTS ....................................... 40
      PING ......................................................... 40
      FTP (FILE TRANSFER PROTOCOL) ............................. 45
   4.2 HEURISTIC RESULTS ............................................ 48
      Objectives .................................................. 48
      Types of Queries used ...................................... 49
Experimental results .......................................................... 50
Importance of Results ......................................................... 58

5  CONCLUSIONS AND FUTURE WORK ............................... 60

5.1 FUTURE WORK ............................................................. 60

A APPENDIX A ................................................................. 61

A.1 DESIGN OF THE SIMULATOR .......................................... 61

B APPENDIX B ................................................................. 68

B.1 DESCRIPTION OF THE ALGORITHM GENERAL (RESPONSE
TIME) ................................................................. 68

THE DATA STRUCTURE OF THIS PROGRAM ....................... 68

THE DESCRIPTION OF THE PROGRAM IN ALGORITHM PARALLEL 70

C APPENDIX C ................................................................. 75

C.1 DESCRIPTION OF THE HEURISTIC .................................. 75

D APPENDIX D ................................................................. 80

D.1 SUMMARY OF RESULTS ............................................... 80

BIBLIOGRAPHY ............................................................... 84

E VITA AUCTORIS ............................................................ 87
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Query Processing System in [AHY79]</td>
<td>5</td>
</tr>
<tr>
<td>Figure 2</td>
<td>An example to calculate response time and total time.</td>
<td>10</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Relations transmitted to QS without reduction</td>
<td>36</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Selection of the best reducers to reduce a Relation</td>
<td>37</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Selection of the best potential reducer</td>
<td>38</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Final schedules for the relations</td>
<td>39</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Structures created and placed in EVENT_LIST</td>
<td>64</td>
</tr>
<tr>
<td>Figure 8</td>
<td>After initial execution of the first structure from the EVENT_LIST queue</td>
<td>65</td>
</tr>
<tr>
<td>Figure 9</td>
<td>The data field of the structure</td>
<td>69</td>
</tr>
<tr>
<td>Figure 10</td>
<td>An example showing how the data structure is stored</td>
<td>70</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Example showing how the schedule for an attribute is constructed</td>
<td>73</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Example showing how the schedule for a relation is constructed.</td>
<td>74</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Characteristics of LAN and WAN.</td>
<td>26</td>
</tr>
<tr>
<td>Table 2</td>
<td>Algorithm of Main Program</td>
<td>32</td>
</tr>
<tr>
<td>Table 3</td>
<td>Algorithm of function find_best_schedule</td>
<td>33</td>
</tr>
<tr>
<td>Table 4</td>
<td>Algorithm of function find_best_reducer_schedule</td>
<td>34</td>
</tr>
<tr>
<td>Table 5</td>
<td>Relation Table</td>
<td>35</td>
</tr>
<tr>
<td>Table 6</td>
<td>Delay Table</td>
<td>35</td>
</tr>
<tr>
<td>Table 7</td>
<td>The output obtained when using the PING.</td>
<td>41</td>
</tr>
<tr>
<td>Table 8</td>
<td>PING results for the site amazon.eng.fau.edu</td>
<td>42</td>
</tr>
<tr>
<td>Table 9</td>
<td>PING results for the site ftp.ipl.rpi.edu</td>
<td>43</td>
</tr>
<tr>
<td>Table 10</td>
<td>PING results for the site sunee.uwaterloo.ca</td>
<td>43</td>
</tr>
<tr>
<td>Table 11</td>
<td>PING results for the site labrea.stanford.edu</td>
<td>44</td>
</tr>
<tr>
<td>Table 12</td>
<td>PING results for the site ftp.cs.uwm.edu</td>
<td>44</td>
</tr>
<tr>
<td>Table 13</td>
<td>FTP results for the site amazon.eng.fau.edu</td>
<td>45</td>
</tr>
<tr>
<td>Table 14</td>
<td>FTP results for the site ftp.ipl.rpi.edu</td>
<td>46</td>
</tr>
<tr>
<td>Table 15</td>
<td>FTP results for the site sunee.uwaterloo.ca</td>
<td>46</td>
</tr>
<tr>
<td>Table 16</td>
<td>FTP results for the site labrea.stanford.edu</td>
<td>47</td>
</tr>
<tr>
<td>Table 17</td>
<td>FTP results for the site ftp.cs.uwm.edu</td>
<td>47</td>
</tr>
<tr>
<td>Table 18</td>
<td>Table showing the correlation factor between PING and FTP</td>
<td>48</td>
</tr>
<tr>
<td>Table 19</td>
<td>Statistical relevance of differences between the heuristic &amp; AHY algorithm</td>
<td>59</td>
</tr>
<tr>
<td>Table 20</td>
<td>Delay Table</td>
<td>64</td>
</tr>
</tbody>
</table>
Table 21  An example showing the DELAY_TABLE .................. 67
Table 22  An example showing the SHORTEST_PATH ............ 67
Table 23  Relation Table ........................................ 72
Table 24  Algorithm of Main Program ............................ 76
Table 25  Function find_best_schedule ......................... 78
Table 26  Function find_best_reducer_schedule ................ 79
Table 27  The table showing the comparisons when unit delays are
considered ......................................................... 80
Table 28  The table showing the comparisons when delays are
changed by 10%. ............................................... 81
Table 29  The table showing the comparisons when delays are
changed by 25%. ............................................... 81
Table 30  The table showing the comparisons taking average
response time when unit delays are considered and
percentage improvement ......................................... 82
Table 31  The table showing the comparisons taking average
response time when delays are changed by 10% and
percentage improvement ......................................... 83
Table 32  The table showing the comparisons taking average
response time when delays are changed by 25% and
percentage improvement ......................................... 83
In the past, storing data in a single large and expensive centralized computer was extensively used in data processing. Due to recent developments in computer networks and workstations, distributed database management systems (DBMS) have become prominent.

The technology of distributed databases is based on two technologies: *computer networks* and *database technology*. In distributed database management systems, the data is distributed and stored at different sites or nodes. These nodes are connected by a computer network. One of the principal objectives involved when retrieving the data is to get the information for a particular query very quickly. Therefore, most of the research now is being carried out in *optimizing query processing* in distributed database management systems so that the data requested by the user is made available quickly.

Computer communication technology has undergone revolutionary changes since the early 80's. Currently, optical communication (fibre optic) technology allows data transmission at very high speeds. Distributed database management systems depend critically on fast and efficient computer communications but current query optimization techniques do not take into account network parameters useful for communication.

One of the main difficulties in a distributed database system is to select an execution strategy that minimizes resource consumption, since equivalent strategies may consume differing amounts of computer resources. Therefore the primary objective is to find the
best strategy to execute a query. Popular methods for query optimization include joins [11, 34], semijoins [14, 13, 12], and improvement strategies [7, 9, 10]. In this thesis we concentrate on semijoin query processing strategies.

1.1 PROBLEM TO BE INVESTIGATED

In computer communication the main task is to send a given piece of information from any source to any destination as fast as possible and at the least cost. Whether the communication will at all take place or not is not an issue in computer communication. In distributed query optimization, in order to find the optimum sequence of semijoins, we clearly need to determine how much is the cost of the semijoin and what is the resulting benefit. Both of these parameters are determined by two factors.

- the amount of data that has to be communicated, and,
- the expected delay from the source to the destination.

The AHY (Apers-Hevner-Yao) query optimization algorithm [2] assumes that the cost to send the same amount of data from one site to another is independent of the address of the source and the destination node. They are only concerned with minimizing the amount of transmission in order to reduce the cost. The fact that the delay in the path used to transmit the relation from the source to the destination may vary widely is ignored in the AHY algorithms.

It is shown here that it is possible to improve the response time — the time elapsed from the initiation to the completion of the query, by developing a query optimization algorithm which takes into account both the network load and the size of the data transmission.
1.2 THE THESIS STATEMENT

*Heuristics for minimizing the response time in distributed query processing can be improved by taking into account the network load.*

1.3 OBJECTIVES AND SCOPE OF THE THESIS WORK

The objectives and the scope of this thesis are as follows:

1. To develop a query optimization heuristic to reduce the response time which takes into account both the network load and the size of the transmission.
2. To compare the new heuristic with the AHY Algorithm GENERAL (Response Time Version).
3. To carry out the simulation experiments under different network load conditions.

1.4 ORGANIZATION OF THE THESIS REPORT

Chapter 2 surveys previous work in two areas related to the thesis. First a brief introduction to distributed query optimization is presented. It also discusses the basic concepts and the benchmark AHY Algorithm GENERAL (Response time version). The second area which is discussed in this chapter is computer networks which facilitate communication in a DDBMS (Distributed Database Management System).

Chapter 3 describes the proposed query optimization heuristic. An example is given to show how the heuristic works. Chapter 4 presents the results of the investigation and the evaluation. Finally chapter 5 presents the conclusions, recommendations and suggests possible future work.
2.1 DISTRIBUTED QUERY PROCESSING AND OPTIMIZATION

A distributed database system (DDS) is a system in which many computers are located at different locations and they are all interconnected by a network. Each of the computers contains a local database system. This collection of databases constitute the global database.

The software which manages the DDS is called the Distributed Database Management System (DDBMS). There are two major components of a DDBMS. The first one is called the user processor and it is responsible for interpreting user input and formatting output, determining the execution strategy (query processor) and monitoring the global transaction execution. The second component is called the data processor and its responsibility is to optimize queries, ensure integrity control, concurrency and to provide recovery should failure occur.

In a DDBMS, the data processor contains four major components that are directly involved in the execution of a distributed query. They are the query processing subsystem, the integrity subsystem, the scheduling subsystem and the reliability subsystem. Efficient and reliable query processing is the result of proper interaction between these four components.
Before a query can be optimized, the \textit{query processing subsystem} must have access to the subset of the database needed to answer the query. This materialization along with the location of the fragments and/or files is provided to the query processing subsystem by the \textit{integrity subsystem}.

The problem of synchronizing update queries on the redundant data in the network is handled by the \textit{integrity subsystem}. It must solve the problem of controlling concurrent
queries so that the database integrity and consistency is maintained while the communication overhead of transmitting control information among the network nodes is minimized.

Once the query processing subsystem receives a materialization for the query, the optimization algorithm (using algorithm PARALLEL, SERIAL, etc) can then produce an optimal distribution schedule. The scheduling subsystem coordinates the various schedules in the strategy result node. The complex distributed strategy will require considerable network coordination of transmission and local processing.

Another objective in a distributed database is to increase the reliability of data in the system. In order to achieve this objective, it is better that a number of copies of the same relation are distributed on different nodes in the network. In the case of a node failure, there is always another node where a copy of the desired data can be found. The reliability subsystem continuously monitors the system for failures. If a failure occurs the reliability subsystem notifies the scheduling subsystem of the event. The scheduling subsystem either waits for the reliability subsystem to integrate the failed component back into the system or it halts execution and requests that the query processing subsystem provide a new schedule based on the current status of the system.

**Distributed Query Processing**

There are a number of problems that are involved in distributed query processing. These problems are extremely complex. The reason for the complexity is that there are so many different variations on

- the kinds of networks available,
- the way in which these networks are set up, and,
- the way in which data is fragmented, replicated and distributed.
Distributed Query processing differs from centralized processing in two significant ways

1. There is a substantial amount of processing delay involved due to the communication among the sites which are involved in the query.

2. There is an opportunity for parallel processing since there are several computers involved in handling the query.

Query processing has been an active area of research ever since the beginning of the development of relational databases. Many different query processing strategies can be employed to optimize the processing of a query. The query entered by the user should be transformed into an equivalent query, so that it can be computed more efficiently. The method of generating this kind of improved strategy while processing a query is called \textit{query optimization}.

The primary objective involved in query optimization is to decide on a strategy for executing each query over the network in the most cost-efficient way.

\textbf{Semijoin Query Processing Strategy}

The simplest distributed query processing strategy is to ship all relations that are involved in a query to the point where the query originated. This strategy is expensive if large relations have to be transmitted to the originating node and if the result of the query only contains a few tuples and attributes. This strategy is referred to as the initial feasible solution (IFS) [2].

Another solution is to use the join as a query processing strategy [11, 16, 43, 34]. This involves joining selected relations before they are transmitted. The join operation is denoted as $\bowtie$. Computing the join of two relations can be expensive. Whenever possible joins should exploit concurrency. Unwanted information is eliminated before
any transmission using reducers. **Reducers** reduce the amount of information that has to be transmitted. It is better to execute the unary operations (selection, projection) as soon as possible.

The projection of a relation R over the set of attributes A is denoted by $\Pi_A[R]$. The result of the projection is obtained by discarding all columns of R that are not in A, and eliminating any duplicate rows if necessary.

The **Semijoin** strategy is an important strategy for query processing. Let $R_1$ and $R_2$ be two relations with common attribute $A$. The **Semijoin** operation is carried out as follows:

1. $R'_1 = \Pi_A[R_1]$
2. Transfer $R'_1$ to the site of $R_2$
3. $R'_1 \bowtie R_2$

The notation $R_1 \bowtie A \rightarrow R_2$ will be used to denote a semijoin operation between the relations $R_1$ and $R_2$ over the common joining attribute $A$.

Semijoins are important in DDS because they usually reduce the amount of data that needs to be transmitted between sites in order to evaluate a query.

In [6], semijoins are used as the principal reduction operator. The paper defines the semijoin operator, explains why semijoins are effective as a reduction operator and presents an algorithm that constructs a cost-effective program of semijoins, given an envelope and a database. An envelope is a relational calculus expression that maps a database into a sub-database[6].

Semijoins are beneficial only when an attribute has a good selectivity [11, 34], in which case the semijoin act as a powerful reducer. In [43] it is stated that although the use
of semijoins reduces the amount of data transfer and is a valuable tool, it is not always superior to the use of joins. The reasons are:

1. Additional messages may be generated when semijoins are employed.
2. For certain networks, the number of messages exchanged rather than the amount of data transferred may be a dominating factor.
3. Many tactics based on semijoin do not take into account local processing costs.

2.2 COST ESTIMATION TECHNIQUES

Cost Measures

The cost function is usually defined in terms of time units or in terms of computing resources such as disk space, disk I/O, buffer space, CPU cost, communication cost and so on. There are two very important performance variables in query optimization, total cost and response time.

**Total Cost**: It is a good measure of resource consumption that will be incurred when processing a query. It is the sum of all the times incurred in processing the operations of the query at various sites and in intersite communication. A general formula for determining the total cost is as follows [34, 11]:

\[
\text{Total Cost} = C_{CPU} \ast \#\text{insts} + C_{I/O} \ast \#I/Os + C_{MSG} \ast \#\text{msgs} + C_{TR} \ast \#\text{bytes}
\]

- \(C_{CPU}\) : is the time to execute one CPU instruction.
- \(C_{I/O}\) : is the time to execute one disk I/O.
- \(C_{MSG}\) : is the fixed time to execute the initiation & receiving of a message.
- \(C_{TR}\) : is the time to execute the transmission of a data unit from one site to another. A typical assumption is that \(C_{TR}\) is a constant.
Response Time: This is the elapsed time for query execution. Since the operations can be executed in parallel at different sites, the response time of a query may be significantly less than its total cost. A general formula for determining the response time is shown below [34, 11].

\[
\text{Response Time} = C_{CPU} \times \text{seq.#insts} + C_{IO} \times \text{seq.#I/Os} + C_{MSG} \times \text{seq.#msgs} + C_{TR} \times \text{seq.#bytes}
\]

Example:

The example illustrates the difference between total cost and response time. In order to compute the answer to a query at site 3, data has to be communicated from site 1(2) to site 3.

![Diagram of site communication](image)

Figure 2: An example to calculate response time and total time.

Assumptions:

1. We only consider the communication costs.
2. \( C_{MSG} \) and \( C_{TR} \) are expressed in time units.

\[
\text{Total Cost} = (C_{MSG} + C_{TR} \times x) + (C_{MSG} + C_{TR} \times y) = 2 \times C_{MSG} + C_{TR} \times (x + y)
\]

\[
\text{Response Time} = \text{Max} \{ C_{MSG} + C_{TR} \times x, C_{MSG} + C_{TR} \times y \}
\]

since the transfers can be done in parallel.

Let \( x = 1000 \) be the number of units to transmit from site 1 to site 3 and \( y = 2000 \) be
the number of units to transmit from site 2 to site 3. If the value of $C_{TR} = 1$, and the value of $C_{MSG} = 20$, then:

$$\text{Total cost} = 2 \times 20 + 1000 \times 1 + 2000 \times 1 = 3040.$$ 

Response Time = max \{ 20 + 1000 \times 1, 20 + 2000 \times 1 \} = 2020 as the transfers are done in parallel.

**Database Statistics**

An important factor affecting the performance of an execution strategy is the size of the intermediate relations that are produced during execution. Therefore, it is necessary to estimate the size of the data transfers. This estimation is based on statistical information about:

1. The base relations
2. Formulae to predict the cardinalities of the results of the relational operations.

The main goal of an optimization algorithm is the production of an optimal query execution strategy. The formulation of such an optimal execution strategy can only be accomplished by an exhaustive search of all the possible strategies. The complexity of such an enumeration is shown to be NP-hard [34] resulting in an algorithm that is too computationally expensive to be of any use. Therefore heuristic algorithms are employed to formulate near-optimal strategies.

As most execution strategies are computed statically [11, 34, 7, 9, 10], they require some means of estimating the various values associated with each relation. These values include:

1. the size of the partial relations.
2. the cardinalities of attributes in reduced relations.
3. the selectivities of attributes in reduced relations.

The selectivity $\rho_i$ [11, 34] is defined as the ratio of the number of distinct attribute values of $R_i.A_j$ to the number of possible attribute values in $A_j$ (in $R_i$).

Let $p_{ai}$ be the probability of a value in an attribute $A$ which appears in Relation $R_i$, $i=1, 2$. Since the values in the relations are independently distributed, the probability that a value appears in both the relations is $p_{1a} \cdot p_{2a}$. Thus the expected number of distinct tuples in common between the two relations is $|A| \cdot p_{1a} \cdot p_{2a}$ where $|A|$ is the cardinality of the domain of the attribute $A$. The estimated size of the reduced relation $R_l$ is $|A| \cdot p_{1a} \cdot p_{2a} \cdot w$ where $w$ is the size of one tuple in bytes.

In a relation which has more than one joining attribute, reducing the relation on an attribute may contribute to the reduction of the attribute values of the associated attributes. Here the computation time in evaluating the equation can be expensive if the number of tuples in the reduced relation is large. In [21], the following equation has been proposed. Let $R_i(A_i)$ and $R_j(A_1A_2)$ be two relations in the semijoin, $R_i - A_1 \rightarrow R_j$ and let $n = |R_j|$, $m = |R_jA_2|$ and $k = |R_jA_2|$ then the factor of reduction of $|R_jA_2|$ is given by

$$p = \begin{cases} 
1 - (1 - k/n)^{m/k} & \text{if } n/m < k \\
1 - (1/n)^k & \text{otherwise}
\end{cases}$$

The above estimation can be extended to multi-attribute semijoins also. It has been shown that a satisfactory estimate can be found for the semijoins using multiple attributes based on the ball color problem [19, 15, 21, 43]. "Given $n$ balls with $m$ colors, find the expected number of colors if $t$ balls are selected from $n$ of the $m$ balls". In this case $n$ represents the number of tuples of $R_i$ before the semijoin, $m$ is the number of distinct values of $R_i$ projected on the $A$ attribute before the semijoin and $t$ is the number of tuples.
of \( R \), after the semijoin. The expected cardinality is

\[
g(m, n, t) = m \cdot \left[ 1 - \prod_{i=1}^{t} \left( \frac{(m(m - 1)/m - i + 1)}{n - i + 1} \right) \right]
\]

While \( t \) is a given parameter in the ball color problem, the number of tuples in \( R \) must
be estimated. In its given form, the application of the formula is computationally
expensive and may result in overflow/underflow errors for large values of \( t \) and a
reasonable approximation for the formula is given by

\[
g(m, n, t) = \begin{cases} 
  m, & t \geq 2m \\
  \frac{(m+t)}{3}, & 2m \geq t \geq (m/2) \\
  (\frac{m}{2}), & t < (m/2)
\end{cases}
\]

It is shown that the cardinality of the relation resulting from the join query can be
found out using a theorem explained in [16].

A **beneficial semijoin** refers to a semijoin in which the benefit of performing the
semijoin exceeds the cost of executing it. The **benefit** of a semijoin can be considered
as the amount of data that is eliminated after the semijoin operation is performed. The
benefit is computed by the following function.

\[
B_{benefit}(R_i - d_{ik} \rightarrow R_j) = s_j - s^1_j \text{ where } s^1_j = s_j \cdot \rho_{ik}
\]

where \( s_j \) is the size of relation \( R_j \) and \( \rho_{ik} \) is the selectivity of attribute \( d_{ik} \).

The **cost** associated with a semijoin refers to the transmission of the joining attribute
after projection from the reducing relation to the reduced relation. Assuming that the
transmission cost is fixed between the sites, the cost of the semijoin can be computed
using

\[
Cost(R_i - d_{ik} \rightarrow R_j) = C_0 + C_1 \cdot b_{ik}
\]

where \( C_0 \) and \( C_1 \) represents the start-up cost for a transmission and the fixed cost per
byte transmitted respectively and \( b_{ik} \) is the size of the attribute.
2.3 SEMIJOIN QUERY OPTIMIZATION STRATEGIES

The aim of query optimization is to produce an optimal strategy. In distributed query optimization, the main problem is to determine a sequence of database operations and data communications such that the cost function is minimized. Here we concentrate on the semijoin strategies for relational databases. In [22], an algorithm based on the query optimization technique of decomposition is developed. This algorithm is implemented in a distributed INGRES database system.

In [2] query optimization algorithms using semijoins to minimize the total time and the response time for simple queries are discussed. The algorithms are discussed in more detail later in this literature.

There are other distributed query optimization strategies. In [28] Kang, H and Roussopolous, N have discussed a two-way semijoin for more cost-effective distributed query processing. The two way semijoin is used to reduce the relations in a more efficient way and can be an efficient reducer like the semijoin operation. The two way semijoin is an extended operator because of the fact that it produces two semijoins from its operands.

If there are two relations, relation $R_i$ and $R_j$ which have a common join attribute $A$, the result of the two way semijoin is a set of two relations $\{ R_i^\prime, R_j^\prime \}$ where $R_i^\prime (R_j^\prime)$ is the projection on the attribute of $R_i(R_j)$ of the join of $R_i$ and $R_j$. The two-way semijoin operation of $R_i$ and $R_j$ is denoted as $R_i \leftarrow A \rightarrow R_j$. We can see that two semijoin operations are performed and they can be represented as,

$$R_i \leftarrow A \rightarrow R_j = \{ R_i - A \rightarrow R_j, R_j - A \rightarrow R_i \}$$

Here $R_i$ and $R_j$ are reduced to $R_i^\prime$ and $R_j^\prime$. 

University of Windsor 1995 14
Simulation results [28] have shown the performance of the two-way semijoin to be more powerful than the ordinary semijoins. A pipeline N-way join algorithm is also proposed using a two-way semijoin [36].

Another semijoin based strategy called the **composite semijoin**, which involves multiple attributes is presented in [35]. A composite semijoin is a semijoin in which the projection and transmission involve multiple columns. In most algorithms, multiple semijoins are performed with common source and destination sites when multiple attributes are involved. Whenever there is a situation like this, it is beneficial to do the semijoins as one composite rather than multiple single column semijoins. Through simulation results, it has been shown [35] that including the possibility of composite semijoins in a query processing algorithm substantially reduces the response time.

Another semijoin strategy [20] uses the hash function, **Hash-semijoin**. Hashing techniques are generally considered to be an efficient way of finding the matching tuples. In [31] Bloom filters are used to filter out the tuples that do not participate in the join. A bloom filter is a large vector of bits that are initially set to zero. This is called **bloom join**.

Consider a join between the relations $R_i$ and $R_j$ with the join attribute $A$. The hash semijoin relation from $R_i$ and $R_j$, denoted by $R_i \overset{h_{ij}}{\rightarrow} R_j$ is defined as

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

where $h_{ij}$ is the hash function associated with the hash semijoin $R_i \overset{h_{ij}}{\rightarrow} R_j$. The implementation of the hash semijoin is done in the following steps,

1. The projection of $R_i$ on the join attribute is done.
2. The join attribute is hashed by $h_{ij}$ and the bit array $B_{ij}$ (i.e. $B_{ij}[k]$ is set to 1 if there exists a join attribute value $v$ in relation $R_i$, such that $h_{ij}=k$).
3. Send $b_{ij}$ to the site of $r_j$, and finally select the tuples of $r_j$, whose join attribute value is $u$ and $B_i[h_i(u)] = 1$, as the result of hash semijoin.

Semijoins with multiple hash functions has been suggested by [40]. Here after step (1), the join attribute is hashed with multiple hash functions and it uses the results as the addresses into the bit arrays respectively. That is, a set of hashing functions $h_1, h_2, \ldots, h_m$ are used, each associated with the bit array $B_1, B_2, \ldots, B_m$, respectively. For each value $v$ of the join attribute, all of the corresponding bits in each $B_i$ must be set (i.e. $B[h_i(v)] = 1, B[h_2(v)] = 1, \ldots, B[h_m(v)] = 1$.) (It is shown in [3] that as we increase $m$, the probability of collisions approach 0) Then these bit-arrays are sent to the site of $r_j$. Each tuple of relation $r_j$, whose join attribute value is $u$, belongs to the semijoin if $B[h_1(u)] = 1, B[h_2(u)] = 1, \ldots, B[h_m(u)] = 1$. Semijoin with multiple hash functions is very complex since many hash functions have to be used to reduce the collisions. Collisions can cause some problems but still hash semijoins are considered useful for the following reasons:

1. **Flexibility**: The cost and benefit of an hash semijoin are affected by the probability of the collisions of the hash function used in this hash semijoin. Since the hash function used in the hash semijoin may be any randomizing function, there may be many hash functions which can be chosen and their probabilities of collisions are all different. Therefore hash semijoins are flexible.

2. **Simplicity**: Since there is only one hash function used in each hash semijoin, there is only one-bit array used in each hash semijoin. The processing overhead and transmission cost of hash semijoins are therefore smaller than that of semijoin with multiple hash functions.
The semijoin query processing strategy has also been extended to fragmented databases. In [14], *domain specific semijoins* are proposed for distributed query processing in horizontal partitioned database systems. A complete description is given in [14].

### 2.4 AHY ALGORITHM

In [2] a family of distributed query optimization algorithms using semijoins to minimize the response time and total time for simple queries (algorithm PARALLEL and algorithm SERIAL) is discussed. *Simple queries* are defined such that at initial local processing, each relation in the query contains only one common join attribute, which is also the only output of the query. A general query that contains multiple join queries is decomposed into simple queries and then the algorithm can be applied to each of them.

It is claimed that algorithm PARALLEL derives minimal response time distribution strategies by searching for cost beneficial data transmissions in the current system state given by \( s_i \), *selectivity* \( p_i \) and schedule response time \( r_i \) of each relation \( R_i \). The *selectivity* \( p_i \) of an attribute is defined as the number of different values occurring in the attribute, divided by the number of all possible values of the attribute. Thus \( 0 < p_i \leq 1 \).

A complete description of the AHY Algorithms is given in [2].

**Algorithm PARALLEL**

The basic strategy of algorithm PARALLEL is to search for cost beneficial data transmissions by trying to join small relations to large relations. First an Initial Feasible solution (IFS) is chosen, where all relations are transmitted in parallel to the query site. The algorithm then tries to improve on the solution by considering alternative schedules where some relations are sent to an intermediate site. The algorithm does not consider all schedules that could be generated for a given relation \( R_i \). Relations larger than \( R_i \)
cannot improve the original schedule of \( R_i \) and thus are discarded after first ordering the relations by increasing sizes after projections on the join attributes. The algorithm PARALLEL is given below:

1. Order the relations \( R_i \) such that \( s_1 \leq s_2 \leq \ldots \leq s_m \) where \( s_i \) is the size of relation \( R_i \).
2. Consider each relation \( R_i \) in ascending order of size.
3. For each relation \( R_j (j<i) \), construct a schedule to \( R_i \) that consists of the parallel transmission of the relation \( R_j \) and all schedules of relation \( R_k (k<j) \). Select the schedule with minimum response time.

**Algorithm SERIAL**

To minimize the total time a serial strategy is discussed in [2]. Given an ordering on the required relations of the simple query, the SERIAL strategy consists of transmitting each relation, starting with \( R_1 \), to the next relation in a serial order. The strategy is represented by \( R_1 \rightarrow R_2 \rightarrow \ldots \rightarrow R_n \rightarrow R_r \), where \( R_r \) is the relation at the result node. There are two cases of the SERIAL strategy. In case 1, \( R_r \) is included in its proper order in the transmission pattern, \( R_1 \rightarrow R_2 \rightarrow \ldots \rightarrow R_r \rightarrow \ldots \rightarrow R_n \rightarrow \ldots \rightarrow R_r \). In case 2, \( R_r \) is not included in its proper order, \( R_1 \rightarrow R_2 \rightarrow \ldots \rightarrow R_n \rightarrow R_r \). It is claimed that the SERIAL strategy has minimal total time when the relations are ordered so that \( s_1 \leq s_2 \leq \ldots \leq s_m \).

The algorithm SERIAL is given below:

1. Order relations \( R_i \) such that \( s_1 \leq s_2 \leq \ldots \leq s_m \).
2. If no relations are at the result node, then select strategy:

\[
R_1 \rightarrow R_2 \rightarrow \ldots \rightarrow R_n \rightarrow \text{result node}
\]
Or else if \( R_r \) is a relation at the result node, then there are two strategies:

\[
R_1 \rightarrow R_2 \rightarrow \ldots \rightarrow R_r \rightarrow \ldots \rightarrow R_n \rightarrow R_r \text{ or }
\]

\[
R_1 \rightarrow R_2 \rightarrow \ldots \rightarrow R_{r-1} \rightarrow R_{r+1} \rightarrow \ldots \rightarrow R_n \rightarrow R_r.
\]

3. Select the one with minimum total time.

Owing to the fact that algorithms PARALLEL and SERIAL only work in specialized situations, the algorithm can be extended to work in general query environments. Algorithm GENERAL (discussed in the next section) is a general heuristic that uses an improved exhaustive search to find efficient distribution strategies for general queries.

**Algorithm GENERAL**

Apers and Hevner developed an improved algorithm called GENERAL to process general queries\(^1\). A general query is characterized by \( \alpha_i \beta_i \geq 1 \) for \( i = 1, 2, \ldots, m \) where \( m \) is the number of relations, \( \alpha_i \) is the number of attributes in relation \( R_i \) and \( \beta_i \) is the number of internodal joining attributes in relation \( R_i \). Let \( \sigma \) represents the number of joining attributes in a query. The tactic used in the algorithm is to reduce the size of a relation with semijoins on different joining attributes.

Each relation \( R_i \) is examined in a small to larger order (i.e., the index of the relation indicates its relative size after projection on the join attribute): \( \text{size}(R_1) \leq \text{size}(R_2) \leq \ldots \text{size}(R_m) \) to find a schedule that has minimum response time or minimum total time, depending upon the declared cost objective.

Algorithm GENERAL makes the following assumptions

1. The cost of processing a query is determined by the transmission cost only.
2. All relations are located at different sites.

---

\(^1\) A general query means that a relation can contain more than one joining attribute.
3. Local processing cost is not taken into account.

4. The query processing strategy is run on a dedicated system.

5. Communication line and subsequent queueing delays are not considered in the algorithm.

6. All initial local processing has been performed.

The Algorithm GENERAL (Response time version) is given below:

1. Do all initial local processing

2. Generate candidate schedules. Isolate each of the $\sigma$ joining attributes, and consider each to define a simple query with an undefined result node.

   • Since response time is being reduced, apply algorithm PARALLEL to each simple query. Save all candidate schedules for integration in step (3).

3. Integrate the candidate schedules. For each relation $R_i$, the candidate schedules are integrated to form a processing schedule for $R_i$. To minimize the response time, procedure RESPONSE is used to integrate the schedules.

4. Remove schedule redundancies. Eliminate relation schedules for relations which have been transmitted in the schedule for another relation.

Procedure RESPONSE is carried out as shown here:

1. Candidate schedule ordering: For each relation $R_i$, order the candidate schedules on the joining attributes $d_{ij}$, $j = 1, 2, \ldots, \sigma$ in ascending order of arrival time. Let $ART_i$ denote the arrival time of candidate schedule $CSCH_i$. (For the joining attributes not in $R_i$, disregard the corresponding candidate schedules)
2. *Schedule Integration* For each candidate schedule $CSCH_i$ in ascending order, construct an integrated schedule for $R_i$ that consists of the parallel transmission of $CSCH_i$ and all $CSCH_k$ with $k<i$. Select the integrated schedules with minimum response time.
2.5 COMPUTER NETWORKS

The connection of a number of autonomous computers that are capable of exchanging information among themselves is called a computer network. The computers should be autonomous because they should be capable of executing programs on their own. The computers in the network are called nodes, sites or hosts. A computer network is a special case of a Distributed Computing Environment (DCE).

Network Topology

Based on the interconnection structure of the computers, we can classify computer networks. There are different topologies which are used. Either point-to-point topology or multi-point topology can be used in the construction of the Wide Area Network. In point-to-point network, every site is connected by an IMP (Interface Message Processors). An IMP is a dedicated processor connected by a communication channel. In a point-to-point network, the IMP's have the responsibility of choosing the path along which a message is transmitted in the presence of alternatives. In a multi-point topology the fundamental medium of transmission is a broadcasting channel such as a radio or satellite link. Another popular form of data communication is Microwave transmission. In a broadcast network, everybody can listen and therefore it is not as secure as a point-to-point network. Most of the wide area networks are implemented according to point-to-point topology.

The different computer architectures used are star, tree and mesh. In a star topology, all the computers are connected to a central computer that coordinates the transmission on the network. Therefore if any two computers want to communicate they have to go through the central computer. One disadvantage is their unreliability. Since the communication between any two computers depends on the availability of the central computer, a failure
at this node will cause the transmission over the network to stop completely. Another disadvantage is the excessive load on the central computer.

In a hierarchy or tree network, all the nodes are connected in the structure of a tree. The transmission among the nodes at the same level has to proceed upward until a common node can be found, and then it proceeds down to the other node. In a meshed interconnection scheme, each node is interconnected to every other node. Even though it provides reliability and better performance, the implementation is very expensive.

**The Physical Transmission of Data**

Due to the fact that the sites in Wide Area Networks are distributed physically over a large geographical area, the communication links are likely to be relatively slow and less reliable as compared with Local Area Networks (LANs). Typical WAN links are telephone lines, microwave links and satellite channels. In LANs all sites are close together, so the communications are of higher speed and have lower error rates than their counterparts in WANs. The most common links are twisted pair, baseband coaxial, broadband coaxial, and fiber optics.

The sites in the computer communication system can be connected physically in a variety of ways. The various topologies are represented as graphs whose nodes correspond to sites. An edge from node A to node B corresponds to a direct physical connection between two sites. In most cases it is just not feasible to connect all sites together. The network topologies like ring or bus network provide for the sharing of transmission facilities among many sites. Therefore the cost incurred by any pair of sites is reduced.

The best known ways to transmit digital information are Wavelength Division Multiplexing (WDM) and Time Division Multiplexing (TDM). In WDM, the frequency spec-
trum is divided into logical channels, with each user having exclusive possession of this frequency band (frequency sharing). In TDM, each user takes turns accessing the entire bandwidth for short periods of time. The advantage of multiplexing schemes is that a large number of voice and data transmissions can occur simultaneously, thus making better use of the existing communication facilities.

Recent advancements in the area of light wave technology, have made it possible to transmit the data on an optical fiber using light pulses to represent bits (a light pulse represents a 1 bit, no light pulse represents a 0 bit). Some reasons why fiber optic technology is so important are that a fiber has high bandwidth, is thin and lightweight, is not affected by electromagnetic interference, and has excellent security because it is nearly impossible to wiretap without detection.

Architecture and Importance of Computer Networks

The Local Area Network (LAN) is defined as a communication network that provides interconnection of a variety of data communicating devices within a small area. LANs support minis, mainframes, terminals and other peripherals. In many cases, these networks can carry not only data but voice, video, and graphics.

The most common type of LAN is the bus or tree using coaxial cable. Rings using twisted pair, coaxial, or even fiber are alternative media. The data transfer rates on LANs are high enough to satisfy most requirements and provide sufficient capacity to permit large numbers of devices to share the network.

The usual length of LANs is less than 3 miles. This restriction is a result of propagation delays in the network.
Metropolitan Area Networks (MANs) are defined as networks that support two way communication over a shared medium, such as optical-fiber cable, span a distance of approximately 50 miles, and may offer point-to-point high speed circuits or packet switched communication. MANs are expected to transmit data at rates of 150 MBits/s. They do not, however have the huge traffic handling capability of a switched exchange network. Essentially a MAN is a very large LAN, using access protocols less sensitive to network size than those used in LANs.

Most early networks were WANs designed for voice communication. Because of the proliferation of computers within individual sites, LANs were developed to interconnect these local computers in order to share data and resources. WANs usually span greater distances and are based on the store-and-forward switching mechanism, and require the routing of packets. LANs on the other hand, span distance of up to 3 miles, are based on the ring topology which constitute a single data link, and are usually error prone.

The characteristics of LANs and WANs [39] are given in the table below

<table>
<thead>
<tr>
<th>Local Area Networks (LAN)</th>
<th>Wide Area Networks (WAN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Within site up to 3 miles</td>
<td>Distances up to thousands of miles</td>
</tr>
<tr>
<td>High Bandwidth (&gt; 1M. bits/s)</td>
<td>Typical data rates up to 100 K. bits/s</td>
</tr>
<tr>
<td>Simpler protocols</td>
<td>Complex protocols</td>
</tr>
<tr>
<td>Interconnect cooperating computers in distributed processing applications</td>
<td>Interconnect autonomous computer systems</td>
</tr>
</tbody>
</table>

Table 1  Characteristics of LAN and WAN. (Continued) . . .
<table>
<thead>
<tr>
<th>Local Area Networks (LAN)</th>
<th>Wide Area Networks (WAN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usually operated by the same organization which operates the computer it connects</td>
<td>May be managed by organizations independent of users.</td>
</tr>
<tr>
<td>Generally digital signalling over private cables</td>
<td>Often use analogue circuits from the telephone system</td>
</tr>
<tr>
<td>Lower error rates (1 in 10^9)</td>
<td>Higher error rates (1 in 10^5)</td>
</tr>
<tr>
<td>Can broadcast a single message to multiple destinations</td>
<td>Generally use point-to-point links</td>
</tr>
<tr>
<td>Common topologies - bus or ring</td>
<td>Common topologies - mesh or star</td>
</tr>
</tbody>
</table>

Table 1 Characteristics of LAN and WAN.

**FTP and PING**

In this thesis the following UNIX network commands have been used:

1. The *ftp* command is the user interface to the Internet Standard File Transfer Protocol (FTP). It is used to transfer files to and from a remote network site. It is described in detail in [1].

2. *PING* is a command [1] which is used to find the round trip delay time for a given pair of sites. It also gives the packet loss statistics for a particular host and can also be used to check if a particular host is active or not. There are a number of options which are available when the PING is used. It is described in detail in [1].
The cost model, the description and an example of the heuristic is discussed in this chapter. Section 3.1 discusses how the cost is measured in this heuristic. Section 3.2 discusses the heuristic in detail and Section 3.3 illustrates an example of this heuristic.

3.1 COST MODEL OF THE HEURISTIC

In this thesis an attempt has been made to minimize the delay involved in communicating all data relevant for the query. In order to do this, a model is needed to determine the delay. The model used here takes into account the fact that there is some time to set up the network before any communication can start. This is the start up (or set-up) time. After that the number of packets containing the data is transmitted from the source node to the destination node. Depending on the delays on each edge of the path from the source to the destination, each packet will take a certain amount of time for communication from the source to the destination. A simple cost model that is used to calculate the delay $\tau$ in all the query processing algorithms is given below:

$$\tau = \sum_{i=0}^{\phi - 1} \text{Cost}^i_{(s \rightarrow d)} + \text{Cost}_{\text{set-up}}$$

where $\text{Cost}^i_{(s \rightarrow d)}$ is the delay to send the $ith$ packet from source $s$ to destination $d$ and $\text{Cost}_{\text{set-up}}$ is the set-up time and $\phi$ is the number of packets to be transmitted from a given source to a destination.
In the AHY algorithm, it is assumed that the time to send a given amount of information from a source to a destination is determined only by the amount of data to be transmitted and that the actual path does not play a role. In other words, they assume that the delay between any source and destination is 1. (i.e., $\text{Cost}_i(s \rightarrow d) = 1$, for all $i$ and for all source-destination pairs).

The PING results were used to find out the delays between a source-destination pair and also for finding the initial set-up times.

The results obtained from the experiments carried out using the PING / FTP (given in Chapter 4) shows that:

- the initial set-up time for any source-destination pair is very small
- the time required to send the packets between sites does not change very much from packet to packet.
- the delay between one source-destination pair on the network is not necessary the same as the delay on another source-destination pair.

As a result of these experiments, the set up time $\text{Cost}_{\text{set-up}}$ is ignored in the heuristic and the following simple cost-model for calculating the delay to send each packet from a given source to a destination has been used:

$$
\tau = \text{Cost}_i(s \rightarrow d) \times \phi
$$

Since a static strategy is used, the contents of the global delay table maintained by the routing manager [39] are used to estimate the $\text{Cost}(s \rightarrow d)$ which is the expected delay/packet from any source node to any destination node. In this approach, when the semijoin optimization strategy is being formulated, the value of $\text{Cost}(s \rightarrow d)$ is determined from the network parameters [39].
The FTP results (given in Chapter 4) give the delays between a source-destination pair to transfer a file of a particular size. Based on the results, the delays used in the heuristic were changed suitably at regular intervals of time.

3.2 ASSUMPTIONS, DEFINITIONS AND NOTATIONS

In this section the assumptions, definitions and the notations used are explained.

The following assumptions are made in the heuristic:

1. Point-to-point communication is used
2. The relations and the query site QS are at distinct sites in the network.
3. SPJ (Select-Project-Join) queries are considered.
4. Local processing costs are ignored.

The assumption that the relations are stored at distinct sites is made because if any two relations exist at the same site then the processing of these relations can be done locally.

In a WAN the sites may be thousands of miles apart and the local processing cost may safely be ignored since it is negligible when compared to the actual communication time. In SPJ queries, the selection and projection operations are done locally before the actual join operations. Therefore in SPJ queries, only the join attributes which participate in the query are considered.

For the query $Q = R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n$ ($n$ is the number of relations involved in the query) to be processed the following information is necessary:

- the Relations $R_1$, $R_2$, ..., $R_n$
- the delays on the different edges of the network
The above information is available in two tables — **Relation table** and **Delay table** as described below.

The following metadata about the relations $R_1, R_2, \ldots, R_n$ is stored in the Relation table:

1. For each relation $R_i, \ i = 1, 2, \ldots, n$
   a. the number of attributes $(m_i)$
   b. the size of each relation $s_i$.
   c. the site where the relation resides, $site_i$

2. For each attribute $d_{ij}, j = 1, 2, \ldots, m_i$ of relation $R_i$,
   a. the selectivity of $d_{ij}, \ \rho_{ij}$
   b. the size (in tuples) of attribute $b_{ij}$.

The Delay table stores the delay to send the packet from a source $s$ to a destination $d$.

Examples of the Relation table and Delay Table are given in Table 12 and Table 13.

**Definition**

If the time to generate and ship the reducer to the site of Relation $R_j$ is less than the reduction in time to send the reduced relation Relation $R_j$ to the query site $QS$ then the semijoin reducer is called **beneficial** for relation $R_j$.

**Example 3.1**

For the Relation table in Table 5, consider using the attribute $d_{12}$ to reduce relation $R_2$. Since there are 100 tuples in attribute $d_{12}$, the time to send $d_{12}$ to $R_2$ is 100 units and the time to send the reduced $R_2$ to Query site $QS$ is $2000 \times 0.2$ where 0.2 is the selectivity of attribute $d_{12}$. The total time of carrying out the operation is 500 units, which is less than
the time required to ship the unreduced relation $R_2$ to $QS$. The reducer $d_{12}$ is beneficial for relation $R_j$.

**Definition**

If relation $R_i$ ($R_j$) has attribute $r_i(r_j)$ defined on the same domain, then $r_i(r_j)$ is a potential reducer for relation $R_j(R_j)$.

**Example 3.2**

For the Relation table in Table 5, the potential reducers of relation $R_2$ are attributes $d_{11}$, $d_{12}$ of relation $R_1$ and attribute $d_{31}$ of relation $R_3$.

**Definition**

If two attributes $r_i$ and $r_j$ are defined on the same domain, then $r_i(r_j)$ is a potential reducer for attribute $r_j(r_j)$.

**Example 3.3**

For the Relation table in Table 5, the potential reducers of attribute $d_{11}$ are attributes $d_{21}$ and $d_{31}$ because they all lie in the same domain.

### 3.3 DESCRIPTION OF THE HEURISTIC

The heuristic is explained in detail here. The objective of our heuristic is to minimize the response time by taking into account

- the amount of data that is being transmitted
- the delay from a source to a destination as given in the Delay Table.

First the time to send the relations from the site where these relations exist to the query site, taking into account the delays on each edge in a path, is calculated. Since the heuristic considered is a greedy heuristic, the relation which gives the worst
communication time is chosen and an attempt is made to minimize this communication
time. A greedy heuristic selects the most beneficial reducer. In this way, the best
execution schedule for each of the relations is generated. The main program is explained
below.

\[
\text{Worst\_Time} = 0 \\
L \rightarrow \text{List of all relations } R_i, 0 \leq i \leq n \\
\text{While } L \text{ is not empty} \\
\{ \\
\text{Choose the relation } R_j \text{ which requires the largest communication time } T_j \text{ to send the} \\
\text{relation to the Query Site QS.} \\
\text{If } T_j < W, \text{ exit from the loop} \\
T_{opt} = \text{find\_best\_schedule}(Q, R_j, W) \\
\text{If } T_{opt} > W, W = T_{opt} \\
\text{Remove the relation } R_j \text{ from List of relations } L \\
\} \\
\text{Print the schedules of all relations}
\]

Table 2 Algorithm of Main Program

First the communication time for each of the relations \( R_i, 1 \leq i \leq n \) to the query site
\( QS \) is computed. The relation \( R_j \) which has got a communication time more than the
communication time of any other relation to the \( QS \) is chosen. Since this communication
time degrades the response time the most, we attempt to reduce this as much as possible
using the cost (communication time to send the reducer) and benefit analysis ( reduction
in delay in communicating needed tuples of \( R_j \) to query site). After the relation \( R_j \) is
chosen, we find out the potential reducers \( r_i, 1 \leq i \leq m, \) (where \( m \) is the number of
potential reducers) for relation \( R_j \). The communication time of sending the potential
reducers \( r_i \) to the relation to be reduced and the communication time of sending the
reduced relation to the query site is calculated. The potential reducer \( r_i \) for relation \( R_j \) which gives the least total communication time is considered.

Next, the potential reducers for the attribute \( r_i \) are determined. For each of the potential reducers \( r_i \), the cost of sending \( r_i \) to \( r_i \) is computed. Using a greedy heuristic the reducer \( r_i \) is further reduced. The sequence of semijoins which reduces the time of sending \( r_i \) to \( R_j \) and the relation \( R_j \) to the query site \( QS \) to the minimum is chosen. This process is repeated for all the potential attributes of relation \( R_j \).

\[
\begin{align*}
\text{Reducer\_List} &\rightarrow \text{List of all reducers } r_i \text{ of relation } R_j \\
\text{For each reducer } r_i \text{ in the list of reducers, reduce relation } R_j \text{ and estimate the time to communicate relation } R_j \text{ to } QS. \\
\text{While Reducer\_List is not empty} \\
&\{ \\
&\quad \text{Choose the reducer } r_i \text{ which gives the best reduced time.} \\
&\quad T_{\text{min}} = \text{find\_best\_reducer\_schedule}(Q, R_j, r_i) \\
&\quad \text{If } (T_{\text{min}} < \text{Time}(R_j \rightarrow Q) \times \text{delay}(R_j, QS)) \text{ then insert this reducer to schedule.} \\
&\quad \text{Remove the reducer } r_i \text{ from the list of reducers Reducer\_List} \\
&\}\n\end{align*}
\]

**Table 3** Algorithm of function find\_best\_schedule

In the function *find\_best\_schedule* (Table 3), first the list of potential reducers for the relation \( R_j \) is computed. The communication time for each of the potential reducers is calculated. The way the communication time is calculated is given below.

\[
C_{\text{time}} = \text{time}_{r_i \rightarrow R_j} + \text{sel}_{r_i} \times \text{size}(R_j) \times \text{Delay}(R_j, QS)
\]

Here \( \text{time}_{(r_i \rightarrow R_j)} \) is the total number of tuples to transmit to relation \( R_j \), \( \text{sel}(r_i) \) is the selectivity of the reducer \( r_i \). The reducer \( r_k \) which gives the best \( C_{\text{time}} \) is picked. After
choosing the best reducer \( r_k \), the next question is whether the time to communicate \( r_k \) can be further reduced by semijoin operations on reducer \( r_k \). This is done by function \textit{find_best_reducer_schedule}. If the particular reducer gives the best improvement in time then this semijoin schedule is appended to the main schedule. The process is carried until all the reducers have been examined. The algorithm for the function \textit{find_best_reducer_schedule} is shown in Table 4.

\begin{table}[h]
\begin{tabular}{|l|}
\hline
\textbf{Reduced_reducer_list} = List of all potential reducers for reducer \( r_i \). \\
Find the communication time for each of the potential reducers \( r_k \), which is used to reduce the reducer \( r_i \). \\
Choose the reducer \( r_k \) which gives the best improvement in time and pass this to the function \textit{find_best_schedule} \\
\hline
\end{tabular}
\caption{Algorithm of function \textit{find_best_reducer_schedule}}
\end{table}

The process is repeated for the relation having the next largest communication time. The process stops when there does not exist a relation which

- has not been examined so far and
- has a communication cost more than the reduced relation having the worst communication time.

A detailed description of the heuristic is given in Appendix C. To illustrate how the algorithm works, an example is considered and explained in the next section.

\section*{3.4 Example of the Heuristic}

The important characteristics of the three relations of query \( Q = R_1 \bowtie R_2 \bowtie R_3 \), are given in Table 5. The example used is taken from [2].
<table>
<thead>
<tr>
<th>Relation</th>
<th>Size</th>
<th>$d_{i1}$</th>
<th>$d_{i2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>1000</td>
<td>400</td>
<td>100</td>
</tr>
<tr>
<td>$R_2$</td>
<td>2000</td>
<td>400</td>
<td>450</td>
</tr>
<tr>
<td>$R_3$</td>
<td>3000</td>
<td>900</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5 Relation Table

In the table the notation $S_i$ represents the size of the relation $R_i$, in terms of the number of packets needed to communicate $R_i$. The relation $R_1$ ($R_2$) consists of two join attributes $d_{11}$ ($d_{21}$) and $d_{12}$ ($d_{22}$). Relation $R_3$ has one join attribute $d_{31}$. Here $b_{ij}$ and $p_{ij}$ stands for the size and the selectivity of the attribute $j$ of relation $i$. The attributes $d_{11}$, $d_{21}$, and $d_{31}$ are defined on the same domain. Similarly the attribute $d_{12}$ and $d_{22}$ are defined on the same domain. Since the proposed heuristic requires the delay information from the network, the following delay table is used.

<table>
<thead>
<tr>
<th>Source</th>
<th>$R_1$</th>
<th>$R_2$</th>
<th>$R_3$</th>
<th>QS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>$R_2$</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>$R_3$</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Table 6 Delay Table

The steps involved in generating the schedule using the heuristic is explained. Initially the worst time is set to 0.

**Step 1**

In the first phase, the communication delays for sending the relations $R_1$, $R_2$, and $R_3$ to the query site QS are shown in Figure 3.
The relation which has the largest communication time is chosen for minimization. Here relation $R_3$ has the largest communication time.

**Step II**

In this step the potential reducers for the relation $R_3$ are determined. The potential reducers for $R_3$, are the attributes $d_{11}$ from $R_1$ and $d_{21}$ from $R_2$. The total time required to transmit the potential reducers to reduce the relation $R_3$ and to transmit the reduced relation $R_3$ to the query site $QS$ are then determined. Since the delay for communicating $d_{21}$ from its site $S_2$ to Site $S_3$ is greater than that for $d_{11}$, the schedule involving $d_{21}$ will have parallel transmission of $d_{11}$ to $S_3$. (Schedule (3) in Figure 4). The communication time to send $d_{11}$ to reduce the relation $R_3$ is less than transmitting the tuples of attribute $d_{21}$ because of the high delay when sending it from $S_2$ to $S_3$. 
Step III

The possibility of reducing the reducer $d_{11}$ is considered. The attributes $d_{11}$, $d_{21}$ and $d_{31}$ are all defined on the same domain and therefore they can be used to reduce one another. When considering the possibility of using $d_{21}$ to reduce $d_{11}$, the cost is the time to communicate $d_{21}$ to Site $S_1$. The size of $d_{21}$ is 400 packets and the delay from $S_2$ to $S_1$ is 1 so the cost is 400 units of time. The total communication time is calculated.

- the size of the attribute $d_{11}$ is reduced to $400 \times 0.4 = 160$ due to the selectivity of attribute $d_{21}$ which is 0.4. The benefit due to the semijoin $d_{21} \times d_{11}$ is $400 \times (1-0.4)$
= 240 packets. The delay in communicating \( d_{21} \bowtie d_{11} \) is reduced by \( 240 \times 5 = 1200 \) units of time.

- to reduce the relation \( R_3 \), the selectivity of \( d_{21} \) and \( d_{11} \) which is \( 0.4 \times 0.4 = 0.16 \) is used, instead of 0.4 of attribute \( d_{11} \) alone. The benefit for communicating the number of tuples of relation \( R_3 \) to the query site is \( 3000 \times 0.4 \times (1-0.4) \). The delay in communicating the reduced relation \( R_3 \) to \( QS \) is \( 720 \times 4 = 2880 \).

Here the cost is less than the benefit and \( d_{21} \bowtie d_{11} \) is a better reducer than \( d_{11} \).

\[
\begin{align*}
\text{RT} : & 400 \times 1 + 400 \times 5 \times 0.4 + 3000 \times 4 \times 0.4 \times 0.4 = 3120 \\
\text{RT} : & 400 \times 2 + 400 \times 7 \times 0.4 + 3000 \times 4 \times 0.4 \times 0.4 = 3840 \\
\end{align*}
\]

Figure 5 Selection of the best potential reducer

**Step IV**

The reducer \( d_{21} \) is considered next. The potential reducers for \( d_{21} \) are \( d_{11} \) and \( d_{31} \) since they are defined on the same domain. If attribute \( d_{11} \) is used to reduce \( d_{21} \) the time to communicate \( d_{11} \) to \( S_2 \) is 400 packets (size of attribute \( d_{11} \) \( \times \) 2 (the delay to from \( S_1 \) to \( S_2 \)) The total cost is 3840. Therefore, \( d_{21} \bowtie d_{11} \) is the best reducer for the
relation $R_3$ and the schedule as shown in Figure 5 is obtained. Therefore the value of Worst_time $W$ is now 3120.

**Step V**

The next relation which gives the largest communication time is $R_2$, since the communication time of sending the relation $R_2$ to the $QS$ is greater than the new Worst time. Therefore, it is important to reduce the relation $R_2$ further. The potential reducers of relation $R_2$ are found. Attributes $d_{11}$, $d_{12}$ of relation $R_1$ and $d_{31}$ of relation $R_3$ are the potential reducers. Then the communication time for each of the potential reducers is calculated. Then similar to relation $R_3$, the best schedule for the relation $R_2$ is constructed. The communication time for the relation $R_2$ is less than the communication time of the worst time (1000 < 3120), so the worst time is not changed.

For the relation $R_I$ the communication time of sending $R_I$ to $QS$ is already less than the worst_time. Therefore it is not necessary to reduce the relation $R_I$ further. The final schedules for the relations $R_1$, $R_2$, and $R_3$ are shown in the figure below.

![Diagram](image)

**Figure 6**: Final schedules for the relations

University of Windsor 1995
4.1 FTP AND PING RESULTS

In order to reflect the changes in the network load, the delays had to be changed at regular intervals. The experiments carried out using the File Transfer Protocol (FTP), gave the time to transfer the files between a source-destination pair and also the delays in the networks during different times of the day. PING was used to find the delays between the source-destination pair and also for finding the initial set-up time. It was also used to check if a particular host was active before the FTP was done. Also in order to use the actual network delays, experiments were carried out for about two months using the File Transfer Protocol (FTP) and PING to check if the results of FTP correlated with the results of PING. The experimental results of using the FTP and PING are given in this section.

PING

In the example shown below, the network host is amazon.eng.fau.edu: 200 represents the number of data bytes that are transferred, 10 represents the count. For the example shown below, we see that it gives the statistics of sending each datagrams (200 bytes); it gives the round trip times for each of the datagrams. Finally it gives the statistics of how many packets were transmitted, the packets that were received and information about packet loss. It also gives the minimum, average and the maximum round trip times for the number of packets specified.
Table 7 The output obtained when using the PING.

For the experiments, five different sites situated across North America were chosen at random. The sites chosen for the experiments are given below:

1. amazon.eng.fau.edu
2. ftp.ipl.rpi.edu
3. sunee.uwaterloo.ca
4. labrea.stanford.edu
5. ftp.cs.uwm.edu

An experiment to find out the time taken by each of the different sites to FTP files of similar sizes was carried out for about two months. Also on the same sites the PING experiments were carried out concurrently. In the PING and FTP results shown below,
eight experiments were randomly selected from the total of 69 experiments. Some PING results are shown below:

<table>
<thead>
<tr>
<th>Number</th>
<th>Time</th>
<th>Day</th>
<th>Date</th>
<th>Initial Set-up Time</th>
<th>(min:avg:max) ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3:30 PM</td>
<td>Monday</td>
<td>Aug 28,'95</td>
<td>3</td>
<td>1/1/3</td>
</tr>
<tr>
<td>2</td>
<td>00:02 AM</td>
<td>Wednesday</td>
<td>Aug 30,'95</td>
<td>7</td>
<td>1/2/10</td>
</tr>
<tr>
<td>3</td>
<td>11:10PM</td>
<td>Saturday</td>
<td>Sept 2,'95</td>
<td>3</td>
<td>1/1/3</td>
</tr>
<tr>
<td>4</td>
<td>11:15 AM</td>
<td>Wednesday</td>
<td>Sept 6,'95</td>
<td>3</td>
<td>1/2/3</td>
</tr>
<tr>
<td>5</td>
<td>12:15 PM</td>
<td>Sunday</td>
<td>Sept 10,'95</td>
<td>3</td>
<td>1/1/3</td>
</tr>
<tr>
<td>6</td>
<td>2:10 PM</td>
<td>Tuesday</td>
<td>Sept 19,'95</td>
<td>13</td>
<td>1/4/18</td>
</tr>
<tr>
<td>7</td>
<td>1:20 PM</td>
<td>Thursday</td>
<td>Sept 21,'95</td>
<td>8</td>
<td>1/3/9</td>
</tr>
<tr>
<td>8</td>
<td>4:15PM</td>
<td>Tuesday</td>
<td>Oct 17,'95</td>
<td>5</td>
<td>1/4/6</td>
</tr>
</tbody>
</table>

Table 8  PING results for the site amazon.eng.fiu.edu
### Table 9 PING results for the site ftp.ipl.rpi.edu

<table>
<thead>
<tr>
<th>Number</th>
<th>Time</th>
<th>Day</th>
<th>Date</th>
<th>Initial Set-up Time</th>
<th>(min/avg/max) ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3:30 PM</td>
<td>Monday</td>
<td>Aug 28,'95</td>
<td>4</td>
<td>1/2/4</td>
</tr>
<tr>
<td>2</td>
<td>00:02 AM</td>
<td>Wednesday</td>
<td>Aug 30,'95</td>
<td>3</td>
<td>1/1/3</td>
</tr>
<tr>
<td>3</td>
<td>11:10PM</td>
<td>Saturday</td>
<td>Sept 2,'95</td>
<td>4</td>
<td>1/1/4</td>
</tr>
<tr>
<td>4</td>
<td>11:15 AM</td>
<td>Wednesday</td>
<td>Sept 6,'95</td>
<td>3</td>
<td>1/2/3</td>
</tr>
<tr>
<td>5</td>
<td>12:15 PM</td>
<td>Sunday</td>
<td>Sept 10,'95</td>
<td>3</td>
<td>1/1/3</td>
</tr>
<tr>
<td>6</td>
<td>2:10 PM</td>
<td>Tuesday</td>
<td>Sept 19,'95</td>
<td>4</td>
<td>1/2/4</td>
</tr>
<tr>
<td>7</td>
<td>1:20 PM</td>
<td>Thursday</td>
<td>Sept 21,'95</td>
<td>3</td>
<td>1/1/3</td>
</tr>
<tr>
<td>8</td>
<td>4:15PM</td>
<td>Tuesday</td>
<td>Oct 17,'95</td>
<td>6</td>
<td>1/2/8</td>
</tr>
</tbody>
</table>

### Table 10 PING results for the site sunee.uwaterloo.ca

<table>
<thead>
<tr>
<th>Number</th>
<th>Time</th>
<th>Day</th>
<th>Date</th>
<th>Initial Set-up Time</th>
<th>(min/avg/max) ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3:30 PM</td>
<td>Monday</td>
<td>Aug 28,'95</td>
<td>4</td>
<td>1/2/4</td>
</tr>
<tr>
<td>2</td>
<td>00:02 AM</td>
<td>Wednesday</td>
<td>Aug 30,'95</td>
<td>3</td>
<td>1/1/3</td>
</tr>
<tr>
<td>3</td>
<td>11:10PM</td>
<td>Saturday</td>
<td>Sept 2,'95</td>
<td>3</td>
<td>1/1/3</td>
</tr>
<tr>
<td>4</td>
<td>11:15 AM</td>
<td>Wednesday</td>
<td>Sept 6,'95</td>
<td>3</td>
<td>2/2/3</td>
</tr>
<tr>
<td>5</td>
<td>12:15 PM</td>
<td>Sunday</td>
<td>Sept 10,'95</td>
<td>18</td>
<td>1/4/21</td>
</tr>
<tr>
<td>6</td>
<td>2:10 PM</td>
<td>Tuesday</td>
<td>Sept 19,'95</td>
<td>3</td>
<td>1/1/3</td>
</tr>
<tr>
<td>7</td>
<td>1:20 PM</td>
<td>Thursday</td>
<td>Sept 21,'95</td>
<td>3</td>
<td>1/2/3</td>
</tr>
<tr>
<td>8</td>
<td>4:15PM</td>
<td>Tuesday</td>
<td>Oct 17,'95</td>
<td>11</td>
<td>1/2/11</td>
</tr>
<tr>
<td>Number</td>
<td>Time</td>
<td>Day</td>
<td>Date</td>
<td>Initial Set-up Time</td>
<td>(min/avg/max) ms</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>------------</td>
<td>---------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>1</td>
<td>3:30 PM</td>
<td>Monday</td>
<td>Aug 28,’95</td>
<td>3</td>
<td>1/1/3</td>
</tr>
<tr>
<td>2</td>
<td>00:02 AM</td>
<td>Wednesday</td>
<td>Aug 30,’95</td>
<td>3</td>
<td>1/2/3</td>
</tr>
<tr>
<td>3</td>
<td>11:10PM</td>
<td>Saturday</td>
<td>Sept 2,’95</td>
<td>14</td>
<td>1/3/14</td>
</tr>
<tr>
<td>4</td>
<td>11:15 AM</td>
<td>Wednesday</td>
<td>Sept 6,’95</td>
<td>3</td>
<td>1/2/3</td>
</tr>
<tr>
<td>5</td>
<td>12:15 PM</td>
<td>Sunday</td>
<td>Sept 10,’95</td>
<td>2</td>
<td>1/1/2</td>
</tr>
<tr>
<td>6</td>
<td>2:10 PM</td>
<td>Tuesday</td>
<td>Sept 19,’95</td>
<td>3</td>
<td>1/2/3</td>
</tr>
<tr>
<td>7</td>
<td>1:20 PM</td>
<td>Thursday</td>
<td>Sept 21,’95</td>
<td>9</td>
<td>1/2/9</td>
</tr>
<tr>
<td>8</td>
<td>4:15 PM</td>
<td>Tuesday</td>
<td>Oct 17,’95</td>
<td>3</td>
<td>1/2/3</td>
</tr>
</tbody>
</table>

Table 11 PING results for the site labrea.stanford.edu

<table>
<thead>
<tr>
<th>Number</th>
<th>Time</th>
<th>Day</th>
<th>Date</th>
<th>Initial Set-up Time</th>
<th>(min/avg/max) ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3:30 PM</td>
<td>Monday</td>
<td>Aug 28,’95</td>
<td>6</td>
<td>1/2/6</td>
</tr>
<tr>
<td>2</td>
<td>00:02 AM</td>
<td>Wednesday</td>
<td>Aug 30,’95</td>
<td>3</td>
<td>1/2/3</td>
</tr>
<tr>
<td>3</td>
<td>11:10PM</td>
<td>Saturday</td>
<td>Sept 2,’95</td>
<td>3</td>
<td>1/1/3</td>
</tr>
<tr>
<td>4</td>
<td>11:15 AM</td>
<td>Wednesday</td>
<td>Sept 6,’95</td>
<td>2</td>
<td>1/1/2</td>
</tr>
<tr>
<td>5</td>
<td>12:15 PM</td>
<td>Sunday</td>
<td>Sept 10,’95</td>
<td>9</td>
<td>1/3/9</td>
</tr>
<tr>
<td>6</td>
<td>2:10 PM</td>
<td>Tuesday</td>
<td>Sept 19,’95</td>
<td>3</td>
<td>1/2/3</td>
</tr>
<tr>
<td>7</td>
<td>1:20 PM</td>
<td>Thursday</td>
<td>Sept 21,’95</td>
<td>3</td>
<td>1/2/3</td>
</tr>
<tr>
<td>8</td>
<td>4:15PM</td>
<td>Tuesday</td>
<td>Oct 17,’95</td>
<td>8</td>
<td>1/3/8</td>
</tr>
</tbody>
</table>

Table 12 PING results for the site ftp.cs.uwm.edu

From the PING experiments, we found out that

- the initial set-up time is very small.
- the delay when sending packets does not change very abruptly over a period of time.
FTP (FILE TRANSFER PROTOCOL)

The ftp command is the user interface to the Internet Standard File Transfer Protocol (FTP). FTP transfers files to and from a remote network site.

For the sites chosen to carry out the experiments, the size of the files in bytes is given below:

1. ftp://amazon.eng.fau.edu – 42106 bytes
2. ftp://ftp.ipl.rpi.edu – 47050 bytes
3. ftp://sunec.uwaterloo.ca – 45407 bytes
4. ftp://labrea.stanford.edu – 49834 bytes

Some ftp results are shown below:

<table>
<thead>
<tr>
<th>Number</th>
<th>Time</th>
<th>Day</th>
<th>Date</th>
<th>FTP (42106 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(secs)</td>
</tr>
<tr>
<td>1</td>
<td>3:30 PM</td>
<td>Monday</td>
<td>Aug 28,'95</td>
<td>35</td>
</tr>
<tr>
<td>2</td>
<td>00:02 AM</td>
<td>Wednesday</td>
<td>Aug 30,'95</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>11:10PM</td>
<td>Saturday</td>
<td>Sept 2,'95</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>11:15 AM</td>
<td>Wednesday</td>
<td>Sept 6,'95</td>
<td>51</td>
</tr>
<tr>
<td>5</td>
<td>12:15 PM</td>
<td>Sunday</td>
<td>Sept 10,'95</td>
<td>11</td>
</tr>
<tr>
<td>6</td>
<td>2:10 PM</td>
<td>Tuesday</td>
<td>Sept 19,'95</td>
<td>43</td>
</tr>
<tr>
<td>7</td>
<td>1:20 PM</td>
<td>Thursday</td>
<td>Sept 21,'95</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>4:15PM</td>
<td>Tuesday</td>
<td>Oct 17,'95</td>
<td>53</td>
</tr>
</tbody>
</table>

Table 13 FTP results for the site amazon.eng.fau.edu
<table>
<thead>
<tr>
<th>Number</th>
<th>Time</th>
<th>Day</th>
<th>Date</th>
<th>FTP (47050 bytes)</th>
<th>(secs)</th>
<th>(kb/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3:30 PM</td>
<td>Monday</td>
<td>Aug 28,'95</td>
<td>18</td>
<td>2.6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>00:02 AM</td>
<td>Wednesday</td>
<td>Aug 30,'95</td>
<td>4</td>
<td>9.5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>11:10PM</td>
<td>Saturday</td>
<td>Sept 2,'95</td>
<td>5</td>
<td>9.3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>11:15 AM</td>
<td>Wednesday</td>
<td>Sept 6,'95</td>
<td>16</td>
<td>2.9</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>12:15 PM</td>
<td>Sunday</td>
<td>Sept 10,'95</td>
<td>10</td>
<td>4.4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2:10 PM</td>
<td>Tuesday</td>
<td>Sept 19,'95</td>
<td>17</td>
<td>2.7</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1:20 PM</td>
<td>Thursday</td>
<td>Sept 21,'95</td>
<td>18</td>
<td>2.6</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4:15PM</td>
<td>Tuesday</td>
<td>Oct 17,'95</td>
<td>36</td>
<td>1.3</td>
<td></td>
</tr>
</tbody>
</table>

Table 14 FTP results for the site ftp.ipl.rpi.edu

<table>
<thead>
<tr>
<th>Number</th>
<th>Time</th>
<th>Day</th>
<th>Date</th>
<th>FTP (45407 bytes)</th>
<th>(secs)</th>
<th>(kb/sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3:30 PM</td>
<td>Monday</td>
<td>Aug 28,'95</td>
<td>16</td>
<td>2.8</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>00:02 AM</td>
<td>Wednesday</td>
<td>Aug 30,'95</td>
<td>3.6</td>
<td>12.7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>11:10PM</td>
<td>Saturday</td>
<td>Sept 2,'95</td>
<td>4</td>
<td>11.3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>11:15 AM</td>
<td>Wednesday</td>
<td>Sept 6,'95</td>
<td>9.4</td>
<td>4.8</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>12:15 PM</td>
<td>Sunday</td>
<td>Sept 10,'95</td>
<td>10.4</td>
<td>4.2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2:10 PM</td>
<td>Tuesday</td>
<td>Sept 19,'95</td>
<td>13.4</td>
<td>3.3</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1:20 PM</td>
<td>Thursday</td>
<td>Sept 21,'95</td>
<td>14</td>
<td>3.2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>4:15PM</td>
<td>Tuesday</td>
<td>Oct 17,'95</td>
<td>11.4</td>
<td>3.9</td>
<td></td>
</tr>
</tbody>
</table>

Table 15 FTP results for the site sunee.uwaterloo.ca
From the tables, it can be seen that the time to transfer a file of a given size varies enormously, depending on

1. source and destination address
2. time of the day.
<table>
<thead>
<tr>
<th></th>
<th>FTP</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PING</td>
<td>site</td>
<td>Correlation factor</td>
</tr>
<tr>
<td></td>
<td>amazon.eng.fau.edu</td>
<td>-0.11204</td>
</tr>
<tr>
<td></td>
<td>ftp.ipl.rpi.edu</td>
<td>-0.03159</td>
</tr>
<tr>
<td></td>
<td>sunec.uwaterloo.ca</td>
<td>0.06414</td>
</tr>
<tr>
<td></td>
<td>labrea.stanford.edu</td>
<td>-0.04851</td>
</tr>
<tr>
<td></td>
<td>ftp.cs.uwm.edu</td>
<td>0.11491</td>
</tr>
</tbody>
</table>

Table 18 Table showing the correlation factor between PING and FTP

From table 18, it was found that the FTP results do not correlate with the PING results. Since the FTP results did not correlate with the PING results, it was not possible to find out the actual delays in the network to formulate the semijoin execution strategy for the heuristic. In the cost model considered in the heuristic, the initial set-up time is ignored, from the experiments carried out using PING, it was found that the initial set-up time is very negligible for any source-destination pair. Also the time taken to send the packets between any source-destination pair does not change abruptly. When the semijoin execution strategy is being formulated using the heuristic, the delays which are present at the time of formulation are considered.

### 4.2 HEURISTIC RESULTS

In this chapter the results and the comparisons between the AHY algorithm and the Heuristic are given.

**Objectives**

The aims of the evaluation were:

- To test the heuristic on a wide variety of select-project-join queries.
To compare the heuristic with the AHY (Apers-Hevner-Yao) algorithm GENERAL (Response Time version).

**Types of Queries used**

For the tests, the database statistics of the relations and the attributes which are only involved in the query after all the local processing is over are considered [4]. Each query had the following characteristics.

1. The query was made up of 3 to 6 relations and the number of join attributes varied from 2 to 4.
2. The domain cardinalities varied between 500 and 1500.
3. The number of tuples for each relation varied from 500–6000.
4. The number of join attributes for in each query ranged from 1 to the maximum number of join attributes, so that the query was “connected” in order to join all the relations to answer a query. The levels of connectivity used for the experiments are 50% and 75%. For the heuristic presented in this thesis, the following conditions had to be satisfied for the different levels of connectivity used:
   a. At least two relations must have an occurrence of the same join attribute
   b. It must be possible to join every relation to form a single conjunctive normal form query.

To run the schedules generated by the AHY algorithm and the heuristic, the simulator is used. A detailed description of the simulator can be found in Appendix A. When formulating the semijoin optimization strategy, the values in the delay table at the time of query optimization are considered. The delays consist of the following characteristics.
1. The delays between the different nodes consisted of values from 0–10. If the delay between any two nodes is zero, this means that there is no direct link between the nodes. To find the shortest path between any source and the destination, the shortest path algorithm is used.

2. In order to reflect the changes in the network load, the delays on the different edges of the network are changed every 100 units of time. The delays considered are changed by 10% and 25%.

**Experimental results**

The heuristics were run on six different test cases. Each run consisted of 1200 queries. For each query, 100 semijoin schedules are constructed using the heuristics and the schedules were run on the simulator. A total of 7200 queries were used to evaluate the performance of the algorithms. The different types of cases run are described below:

- Unit delays are considered and run on the simulator with a 10% delay change and under 50% connectivity.
- Delays from the delay table at the time of query optimization are used and run on the simulator with 10% delay change and under 50% connectivity.
- Delays from the delay table at the time of query optimization are used and run on the simulator with 25% delay change and under 50% connectivity.
- Unit delays are considered and run on the simulator with 10% delay change and under 75% connectivity.
- Delays from the delay table at the time of query optimization are used and run on the simulator with 10% delay change and under 75% connectivity.
- Delays from the delay table at the time of query optimization are used and run on the simulator with 25% delay change and under 75% connectivity.

The complete data summaries for each run are given in Appendix D.

The observations of the results are given below:

1. When unit delays were considered, the heuristic performed well compared to the heuristic AHY algorithm for about 55%-75% of the queries irrespective of the query type.

2. When the delays from the delay table at the time of query optimization are used, the heuristic performed well compared to the AHY algorithm for about 70%-80% of the queries irrespective of the query type.

3. In all the cases, for about 10%-25% of the queries, the heuristic and the AHY algorithm were equal.

4. Irrespective of the query type, the average improvement of the heuristic compared to the AHY algorithm is between 10%-15% [Appendix D].

The tables shown below show the average response times of AHY algorithm (Response time version) and the Heuristic for each query type for the different cases. Here low indicates that unit delays are considered, med (medium) indicates that the differences in delays are changed by 10%, and high indicates that the differences in delays are changed by 25%.
Importance of Results

The procedure t-tests using the Statistical Analysis System (SAS) were run on all the different types of queries. The mean response times of the new heuristic is less than the AHY algorithm (Response time version) in all the cases. Therefore the new heuristic is better than the AHY algorithm (Response time version). Since the AHY algorithm outperforms the proposed heuristic in 5%-15% of the queries, and also in 10%-15% of the queries both the algorithms give the same response times, it is not surprising to find that the percentage reduction was not found to be significant.

In the table, the runs in which it was not possible to disprove the null hypothesis are indicated by a X. We can see clearly that the proposed heuristic works very well

- when the selectivity is above 50% and the delays in the network are high
- when the selectivity is above 75% and the delays in the network are high

Therefore the results are only significant when the differences in delays are changed by 25% in the network. We also see that it is significant in only some cases when the differences in delays are changed by 10%.
<table>
<thead>
<tr>
<th>Query</th>
<th>50N-high</th>
<th>75N-high</th>
<th>50N-med</th>
<th>75N-med</th>
<th>50N-low</th>
<th>75N-low</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-2</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3-2</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3-4</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4-2</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4-3</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4-4</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5-2</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5-3</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5-4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6-2</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6-3</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6-4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 19  Statistical relevance of differences between the heuristic & AHY algorithm
CHAPTER 5
CONCLUSIONS AND FUTURE WORK

There are many possible strategies for processing queries, especially complex queries, and a substantial amount of time and effort is needed to select an optimal strategy. In the earlier distributed DBMSs the objective was to minimize the transmission costs in terms of the message size. In this thesis a new heuristic which takes into account both the network load and also the size of the data to be transmitted is presented. The heuristic uses a static strategy and takes the delays which are present at the time of formulating the semijoin execution strategy. From the results, it is shown that the new heuristic is comparatively efficient and cost effective. In the heuristic, if the worst time is greater than the time to communicate the relations to the query site, then those relations are not considered for further reductions since it is futile. It is important to point out that the AHY outperforms this heuristic for about 5%-15% of the queries.

FUTURE WORK

Finding the actual network delays was not possible using the PING and the FTP. Therefore work can be done to provide the actual network delays between any source-destination pair maintained by the network manager.
APPENDIX A

DESIGN OF THE SIMULATOR

In this thesis, a network simulator was developed which was written in C programming language. The response time is calculated using this simulator. The description of this simulator is described in this appendix. The delays are changed at regular intervals of time to reflect the changes in the network load. Here we just assume a point-to-point network. The input to this simulator is the schedule from the AHY algorithm and the heuristic. The output of this simulator is the response time for each of the schedules.

The simulator takes the schedules which are generated by the heuristics in the form of an adjacency list. The term schedule represents the sequence of transmissions which is generated using the AHY algorithm or our Heuristic. An example of an adjacency list is shown below.

1 0 1 1 1 --> 3 630 3 1 2 --> NULL
2 1 3 3 1 --> 2 187 2 -1 7 --> NULL
3 0 3 3 3 --> 1 290 1 3 4 --> NULL
4 1 1 1 3 --> 2 127 2 -1 7 --> NULL
5 0 2 2 2 --> 1 160 1 2 6 --> NULL
6 1 1 1 2 --> 2 92 2 -1 7 --> NULL
7 3 2 3 -1 --> 5 43 -1 -1 0 --> NULL

It indicates the physical_id, the indegree, the source site, the relation number and attribute number of the attribute; if a relation is being considered the attribute number is represented as -1. Then it indicates the destination site, the number of packets that
are being transmitted, the relation number and the attribute number (the attribute or the relation being reduced) at the destination and the logical_id. The corresponding schedule for the adjacency list is shown below.

\[
\begin{array}{c|c|c}
\text{d}_{11} & 430 & \text{d}_{31} & 187 \\
\text{d}_{33} & 290 & \text{d}_{13} & 127 \\
\text{d}_{22} & 160 & \text{d}_{12} & 92 \\
\end{array}
\]

R$_2$ 43 QS

In the program, first the indegree of the schedules are checked, the schedules which have indegree zero are considered and a separate structure is constructed. The generate function is used to generate the packets from a given source that have to be transmitted to the destination; the receive function is used to confirm that the packet which is transmitted has arrived at the destination. A global event queue called EVENT_LIST is maintained, where each node is stored and processed one at a time. A complete description of each of the functions used in the simulator is explained below. Also an example is shown explaining how the simulator actually works.

It consists of the following four functions.

1. **Check_indegree**: This function takes the input in the form of an adjacency list from the AHY algorithm or heuristic and checks the indegrees of all the elements; whichever has the indegree zero is picked up from the list and a node is created with current_time+1, pointer to the function generate and the data (the source, destination
and the number of packets that have to be transmitted). This is inserted into the EVENT_LIST.

2. *Generate*: This function creates two structures and inserts them into the queue EVENT_LIST. It creates the first structure with the current_time + the delay from the source to the destination, a pointer to the function receive, and the data (the source, destination and the number of packets). The packets will be the number of packets that are remaining to be received at the destination. It also creates another structure only if there are packets which are remaining to be transmitted, with current_time + the round trip delay from source to destination, pointer to the function generate, and the data (the source, destination and the number of packets remaining to be transmitted). When considering the delays, the round trip delay is considered assuming that the packets are transmitted from the source to the destination without any loss of packets.

3. *Receive*: This function first checks the number of packets. If the number of packets is zero, then it checks the destination node with the source node in the adjacency list and also it checks if the logical_id is the same as the physical_id. If both are equal it reduces the indegree by 1. Later it checks the indegree of the source_node. If the indegree is zero then it creates a structure with current_time + 1, (After all the packets have been transmitted, for the next semijoin schedule to be executed, we assume that the new packet is generated after 1 unit of time), pointer to a function generate, the data (the source, destination and the number of packets to be transmitted). This structure is inserted into the queue EVENT_LIST.

4. *Process_Event_List*: This is the main function which processes the elements which
are stored in the EVENT_LIST. It picks up the first node from the EVENT_LIST and then checks the function that has to be processed; if it is generate, then the generate function is called; otherwise, the receive function is called. When all the elements in the EVENT_LIST are processed, the transmission time is output.

To illustrate how the simulator works, the example for the given adjacency list is shown below. Since the delay values are required, the following delay table is considered.

<table>
<thead>
<tr>
<th>Source</th>
<th>R₁</th>
<th>R₂</th>
<th>R₃</th>
<th>QS</th>
</tr>
</thead>
<tbody>
<tr>
<td>R₁</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>R₂</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>R₃</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Table 20 Delay Table

Step 1: The indegrees are checked by scanning through the entire list. There are three schedules with indegree zero. Therefore the structures as shown in the figure below is created.

![Figure 7 Structures created and placed in EVENT_LIST](image-url)

64
Here the current time is taken as zero, therefore the time for each of the structures is taken as 1. The structure is stored as the EVENT_LIST queue.

**Step 2**: The execution of a schedule starts after a warm-up time. Then the actual simulation takes place, that is the *process_event_list* starts. It takes the first structure from the queue and checks the function which has to be carried out. Since it is the generate function, the generate function is executed and the two structures are created and inserted in the EVENT_LIST as shown in the figure below.

![Image of structures](image)

*Figure 8 After initial execution of the first structure from the EVENT_LIST queue*

The structures created are the structures with the function generate, with the delay from the source 1 to destination 3, and the structure with the function receive with the round trip delay from the source 1 to destination 3. When all the packets from the given source to a destination have been transmitted, the receive function checks the logical_id and the physical_id. Consider the case, when all the 160 packets have been transmitted from source 2 to destination 1; then the receive function checks if the logical_id (6), i.e. the attribute being reduced (relation_number and attribute_number, \(d_{12}\)) is the same as the physical_id on the left hand side in the adjacency list. If both are same then the indegree
is reduced to one. For the example considered, the indegree where the physical_id is 6 is reduced to zero.

Before reducing the indegree:
6 1 1 1 2 --> 2 92 2 -1 7 --> NULL

After reducing the indegree:
6 0 1 1 2 --> 2 92 2 -1 7 --> NULL

If the indegree is zero then the receive function creates a structure and places it in the EVENT_LIST. In other words, the indegree represents the expected number of transmissions at the destination node.

There are many algorithms to handle the routing of a packet from one node to another. In this simulator, the adaptive Dijkstra’s Algorithm, which can compute the shortest path, is used to transmit the packets from one node to another. This algorithm consists of two routing techniques. One is adaptive and the other is the shortest path routing. The term adaptive means that the algorithm attempts to change its routing decisions to reflect changes in the current traffic. A detailed description of this algorithm can be found in [39].

In this simulator, the shortest path between two nodes is based on recent changes in the network traffic. The program has an array “DELAY_TABLE” which stores the delays from all the source and the destination nodes. This array is updated at regular intervals. Table below is an example of a DELAY_TABLE. Dijkstra’s algorithm uses this array to determine the routing. The results of these calculations are stored in another array called “SHORTEST_PATH”. This array allows us to determine the shortest path between any two nodes in the network. In this array, the delays are stored between every source and
destination without considering which route it takes for the packet to be transmitted.

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2 0 2 4</td>
</tr>
<tr>
<td>2</td>
<td>4 4 0 1</td>
</tr>
<tr>
<td>3</td>
<td>2 - 2 3</td>
</tr>
<tr>
<td>4</td>
<td>3 3 - 4</td>
</tr>
<tr>
<td>5</td>
<td>1 3 2 -</td>
</tr>
</tbody>
</table>

Table 21 An example showing the DELAY_TABLE

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>- 6 3 7 4</td>
</tr>
<tr>
<td>2</td>
<td>4 - 7 4 1</td>
</tr>
<tr>
<td>3</td>
<td>2 4 - 6 5</td>
</tr>
<tr>
<td>4</td>
<td>3 5 4 - 4</td>
</tr>
<tr>
<td>5</td>
<td>1 3 6 4 -</td>
</tr>
</tbody>
</table>

Table 22 An example showing the SHORTEST_PATH
APPENDIX B

DESCRIPTION OF THE ALGORITHM GENERAL
(RESPONSE TIME)

This appendix describes our implementation of the AHY algorithm. The data structure, the input parameters and the output parameters are discussed.

THE DATA STRUCTURE OF THIS PROGRAM

The schedule for a particular relation is stored as an n-ary binary tree. Each structure consists of ten items. These items are:

- the relation number
- the attribute number
- the original size of the relation or attribute
- the selectivity of the attribute considered
- the changed size (size after a relation or attribute is reduced)
- the indegree
- the reduced relation number
- the reduced attribute number
- a pointer to the child list
- a pointer to the sibling list

For the attribute or the relation which is reduced after the semijoin, the reduced number and the attribute number are also included in the structure. If the relation is being reduced, then the attribute number is represented as —1. The indegree gives the value of
the number of transmissions. If the indegree is one, then it means there is a transmission before this one. Figure below shows an example of the segment.

Figure 9 The data field of the structure
For the given schedule shown in figure 1, the corresponding figure 2 shows how the schedule is stored using the data structure. If any two attributes are sent in parallel to the site where the relation or attribute resides, then they are stored as siblings, here segment B, E and F are siblings and are the children of parent A, and also C and D are the children of B and are siblings.

**THE DESCRIPTION OF THE PROGRAM IN Algorithm PARALLEL**

This program is based on Algorithm GENERAL. The following are the steps of this
program.

1. Perform the following on each of the common join attributes in the query in order to generate the schedule for each of the attributes.

   a. The first schedule is to send the attribute directly to the query site.

   b. Find the schedule with the smallest response time.

   c. Find the next smallest common join attribute and do the semijoin with the previous schedule.

   d. Repeat step 1c until all of that attributes are considered.

   e. Find the schedule with minimum response time from step 1c as the result.

2. Perform the following on each of the relations in order to generate a schedule for each of the relations.

   a. Collect the valid common join attributes from step 1. The term “valid common join” attribute means that:

      • If relation \( R_i \) is processed, then all the common join attributes with the relation number \( i \) will not be considered.

   b. Order the schedules which are generated in step 2a in ascending order.

   c. Find the schedule with the minimum response time from step 2b and do the semijoin with the current relation.

   d. Find the schedule with the next smallest response time from step 2b and do the semijoin with the previous schedule.

   e. Repeat step 2d until all the valid attributes of the current relation are considered.
f. Find the schedule with the minimum response time from step 2d as the result for the current relation.

The following relation table is the example for the illustration of Algorithm PARALLEL.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Size</th>
<th>$d_{i1}$</th>
<th>$d_{i2}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$:</td>
<td>1000</td>
<td>400</td>
<td>100</td>
</tr>
<tr>
<td>$R_2$:</td>
<td>2000</td>
<td>400</td>
<td>450</td>
</tr>
<tr>
<td>$R_2$:</td>
<td>3000</td>
<td>900</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 23 Relation Table

The complete result can be found in the example of Algorithm PARALLEL in Section 2.4.

To find the minimum time for the attributes, attributes $d_{11}$, $d_{21}$ and $d_{31}$ which are in the same domain are considered. The attributes are arranged in increasing order of size. The schedule for attribute $d_{11}$ is constructed by sending $d_{11}$ to the query site. This schedule is placed in the ATTRIB_SCHEDULE_LIST. The next smallest attribute $d_{21}$ is chosen and a schedule is constructed by sending it to the query site; also another schedule is constructed by sending $d_{11}$ to reduce the attribute $d_{21}$. The schedule with the minimum response time is determined and placed in the ATTRIB3_SCHEDULE_LIST. In a similar way the schedules for the attribute $d_{31}$ are constructed. The one which has got the minimum response time is chosen and placed in the list ATTRIB_SCHEDULE_LIST. In a similar way the schedules with minimum response times for the attributes $d_{12}$ and $d_{22}$ are found and placed in the list.

72
For each of the relations being considered, the valid common join attributes are considered and the schedules for these attributes are picked from the ATTRIB_REDUCER_LIST. These schedules are stored in ascending order of response time in RELATION_REDUCER_LIST. The schedule for each attribute is constructed by doing the semijoin with the current relation. Each of the schedules considered is stored in the SCHEDULE_LIST. The schedule which gives the minimum response time is determined and is placed in the list FINAL_SCHEDULE. The program stops after
all the relations have been examined. The figure below shows how the schedule with minimum response time is chosen for relation $R_3$.

![Diagram showing schedule list for $R_3$]

Figure 12. Example showing how the schedule for a relation is constructed.
APPENDIX C

DESCRIPTION OF THE HEURISTIC

The function find_best_schedule will create a schedule and try to reduce the time to communicate the relation $R$ to query site $QS$ to a value less than Worst_time $W$. If that is possible, the function will not attempt any more optimization. Otherwise it will try to reduce the time to communicate relation $R$ to $QS$ as much as possible. The function returns the estimated time to communicate $R$ to $QS$.

First the list of potential reducers for the relation being reduced is found. For each of the potential reducers the communication time is calculated. The way the communication time is calculated is given below.

$$C_{time} = time_{r_i \rightarrow R_j} + sel_r \times size(R_j) \times Delay(R_j, QS)$$

Here $time (r_i \rightarrow R_j)$ is the sum of the

- delay to send all tuples in the reducer $r_i$ the site of $R_j$
- delay to send reduced $R_j$ to query site

The reducer $r_i$ which gives the minimum communication time is chosen.
Given a Relation Table which contains the size of Relations, $S_i$, site where the relation exists $Site_i$, $p_i$, and $d_i$ for the jth attribute of relation i.

Given a $\text{DELAY}[1..\text{NUM\_NODE}, 1..\text{NUM\_NODE}]$

{Compute the communication time for each relation}

For $i = 1$ to $\text{NUM\_REL}$ do

   $\text{Communication\_Time}[i] = \text{size}[i] \times \text{DELAY}[\text{site}[i], QS]$

{Find the relation which has the largest communication time}

   $\text{max} = \text{Communication\_Time}[1]$

For $j = 1$ to $\text{NUM\_REL}$ do

   if ( $\text{max} < \text{Communication\_Time}[j]$)
      
      $\text{max} = \text{Communication\_Time}[j]$
      
      $\text{num} = j$

{initialise the boolean check_rel for all relations as FALSE}

For $i = 1$ to $\text{NUM\_REL}$ do

   $\text{check\_rel}[i] = \text{FALSE}$

$\text{Worst\_time} = 0$;

$\text{check\_rel}[\text{num}] = \text{TRUE}$

$\text{time} = \text{max}$

While $\text{NUM\_REL} <> 0$

   if($\text{time} < \text{Worst\_time}$)

      $\text{check\_rel}[\text{num}] = \text{FALSE}$

      Calculate the schedules of the other relations.

      Insert the data associated of relation $\text{num}$ in $\text{SCHEDULE}$

      $\text{response\_time} = \text{find\_best\_schedule}(QS, \text{num}, \text{Worst\_time})$;

      if ( $\text{response\_time} > \text{Worst\_time}$)

         $\text{Worst\_time} = \text{response\_time}$

      Remove the relation $\text{num}$ from the list of relations

Print the final schedules of the relations

Table 24 Algorithm of Main Program

This reducer is passed to the function $\text{find\_best\_reducer\_schedule}$. The function $\text{find\_best\_reducer\_schedule}$ checks whether the communication time of the reducer $r_i$
may be reduced further. The potential reducers for the reducer are found out and stored in REDUCER_LIST. Then the communication time for each of the reducer in REDUCER_LIST is calculated by sending the reducer to the $r_i$. Also we calculate the time by sending the different reducers in parallel to reduce $r_i$. The reducer which gives the best improvement in time (the communication time is less than the worst time) is chosen and passed back to the function $\text{find\_best\_schedule}$. If the time is less than the communication time of sending only $r_i$ to $R_j$ and $R_j$ to QS, then it is appended to the schedule of relation $R_j$.

In a similar way, the next reducer which gives the best improvement in time is found out and the process is carried out until all the reducers have been examined. The function uses semijoins, to reduce, to a value less than time $W$, the time to communicate the needed tuples of $R_j$, to QS. As soon as the objective of reducing the communication time to below $W$ is realized, the function does not attempt further optimization since such optimization is futile.

The process is repeated for the relation having the next largest communication time. The process stops when there does not exist a relation which

- has not been examined so far and
- has a communication cost more than the reduced relation having the worst communication time.
function find_best_schedule ( Query site QS, relation R, Worst_time W)

{Find the number of potential reducers for relation r}

numReducers = 0;

For i = 1 to NUM_ATTR

    if ( b[R][i] != 0) { to check if attribute size is not equal to 0)

        For j = 1 to NUM_REL

            if ( j != num and b[j][i] != 0) { to check not the same relation

                or the same attribute }

{Compute the communication time for each reducer }

For i = 1 to numReducers

    time[i] = Size[reducer] * DELAY[site[reducer], site[R] + Size[R] * selectivity[reducer] * DELAY[site[R], QS]

Find the reducer which gives the minimum communication time, time[reducer_num].

best_time[i] = find_best_reducer_schedule(QS, r, reducer_num, Worst_time)

If ( best_time[i] < (Size[R] * DELAY[site[R], QS])

    Insert this schedule to SCHEDULE of relation R

    time[reducer_num] = 99999999;

    improvement_possible = 1;

While (improvement_possible)

    For i = 1 to numReducers

        next_best = 99999990

        if ( next_best > time[i]) next_best = i

        if ( next_best = 99999990) improvement_possible = 0;

        else

            best_time = find_best_reducer_schedule(QS, R, i, W)

            Check if new_reducer gives the best improvement in time

            If its best, insert schedule of this reducer to SCHEDULE.

            time[next_best] = 99999999

}

Table 25 Function find_best_schedule.
function `find_best_reducer_schedule (QS, relation, reducer, Worst_time)`
This function checks if the communication time / selectivity of reducer may be further reduced by some other semijoin. An attribute $r_j$ is said to be a reducer of attribute $r_i$, if they are defined on the same domain.

{ find the potential reducers for the reducer `reducer` }
For $i = 1$ to $\text{NUM}_\text{REL}$
    If ($b[i][\text{reducer}] != 0$ and $i != \text{relation}$) { not the reducer being considered and attribute size is not zero}
        Get the potential reducers and store in REDUCER_LIST
Calculate the communication_time for each reducer in REDUCER_LIST.
Also calculate the communication time by sending the reducers parallely to the `reducer`
Consider the reducer[reducers] which gives the best improvement in time
Pass the new communication time to `find_best_schedule`

Table 26 Function `find_best_reducer_schedule`


APPENDIX D

SUMMARY OF RESULTS

The tables below shows the number of times the heuristic showed an improvement over AHY algorithm, and, the number of times the response time was equal, for all the query types and the different cases.

<table>
<thead>
<tr>
<th>QUERY_TYPE</th>
<th>50N(100 Queries)</th>
<th>75N(100 Queries)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AHY</td>
<td>EQUAL</td>
</tr>
<tr>
<td>3-2</td>
<td>16</td>
<td>22</td>
</tr>
<tr>
<td>3-3</td>
<td>22</td>
<td>20</td>
</tr>
<tr>
<td>3-4</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>4-2</td>
<td>19</td>
<td>14</td>
</tr>
<tr>
<td>4-3</td>
<td>21</td>
<td>15</td>
</tr>
<tr>
<td>4-4</td>
<td>18</td>
<td>14</td>
</tr>
<tr>
<td>5-2</td>
<td>12</td>
<td>21</td>
</tr>
<tr>
<td>5-3</td>
<td>15</td>
<td>22</td>
</tr>
<tr>
<td>5-4</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>6-2</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>6-3</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>6-4</td>
<td>15</td>
<td>16</td>
</tr>
</tbody>
</table>

Table 27 The table showing the comparisons when unit delays are considered
<table>
<thead>
<tr>
<th>QUERY_TYPE</th>
<th>50N(100 Queries)</th>
<th>75N(100 Queries)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AHY</td>
<td>EQUAL</td>
</tr>
<tr>
<td>3-2</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>3-3</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>3-4</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>4-2</td>
<td>8</td>
<td>19</td>
</tr>
<tr>
<td>4-3</td>
<td>7</td>
<td>16</td>
</tr>
<tr>
<td>4-4</td>
<td>8</td>
<td>20</td>
</tr>
<tr>
<td>5-2</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>5-3</td>
<td>6</td>
<td>19</td>
</tr>
<tr>
<td>5-4</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>6-2</td>
<td>6</td>
<td>12</td>
</tr>
<tr>
<td>6-3</td>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>6-4</td>
<td>8</td>
<td>18</td>
</tr>
</tbody>
</table>

Table 28 The table showing the comparisons when delays are changed by 10%.

<table>
<thead>
<tr>
<th>QUERY_TYPE</th>
<th>50N(100 Queries)</th>
<th>75N(100 Queries)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AHY</td>
<td>EQUAL</td>
</tr>
<tr>
<td>3-2</td>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>3-3</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>3-4</td>
<td>9</td>
<td>18</td>
</tr>
<tr>
<td>4-2</td>
<td>8</td>
<td>19</td>
</tr>
<tr>
<td>4-3</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>4-4</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>5-2</td>
<td>6</td>
<td>16</td>
</tr>
<tr>
<td>5-3</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>5-4</td>
<td>7</td>
<td>12</td>
</tr>
<tr>
<td>6-2</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>6-3</td>
<td>7</td>
<td>19</td>
</tr>
<tr>
<td>6-4</td>
<td>7</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 29 The table showing the comparisons when delays are changed by 25%.

81
The tables 26, 27 and 28 shows the average response times of the heuristic and the AHY algorithm. The percentage reduction in the response time using the heuristic compared to the AHY algorithm is also given, for all the query types and for the different cases.

<table>
<thead>
<tr>
<th>QUERY_TYPE</th>
<th>50N(100 Queries)</th>
<th>75N(100 Queries)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AHY</td>
<td>HEURISTIC</td>
</tr>
<tr>
<td>3-2</td>
<td>7518</td>
<td>6427</td>
</tr>
<tr>
<td>3-3</td>
<td>8948</td>
<td>7413</td>
</tr>
<tr>
<td>3-4</td>
<td>7348</td>
<td>6349</td>
</tr>
<tr>
<td>4-2</td>
<td>7183</td>
<td>6276</td>
</tr>
<tr>
<td>4-3</td>
<td>7356</td>
<td>6445</td>
</tr>
<tr>
<td>4-4</td>
<td>6403</td>
<td>5761</td>
</tr>
<tr>
<td>5-2</td>
<td>5314</td>
<td>4526</td>
</tr>
<tr>
<td>5-3</td>
<td>5884</td>
<td>5053</td>
</tr>
<tr>
<td>5-4</td>
<td>3969</td>
<td>3499</td>
</tr>
<tr>
<td>6-2</td>
<td>6190</td>
<td>5546</td>
</tr>
<tr>
<td>6-3</td>
<td>4789</td>
<td>4045</td>
</tr>
<tr>
<td>6-4</td>
<td>4446</td>
<td>3877</td>
</tr>
</tbody>
</table>

Table 30: The table showing the comparisons taking average response time when unit delays are considered and percentage improvement.
<table>
<thead>
<tr>
<th>QUERY_TYPE</th>
<th>50N(100 Queries)</th>
<th>75N(100 Queries)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AHY</td>
<td>HEURISTIC</td>
</tr>
<tr>
<td>3-2</td>
<td>5806</td>
<td>5154</td>
</tr>
<tr>
<td>3-3</td>
<td>8106</td>
<td>7040</td>
</tr>
<tr>
<td>3-4</td>
<td>6304</td>
<td>5633</td>
</tr>
<tr>
<td>4-2</td>
<td>7786</td>
<td>6904</td>
</tr>
<tr>
<td>4-3</td>
<td>8008</td>
<td>7036</td>
</tr>
<tr>
<td>4-4</td>
<td>6736</td>
<td>5974</td>
</tr>
<tr>
<td>5-2</td>
<td>7398</td>
<td>6540</td>
</tr>
<tr>
<td>5-3</td>
<td>5666</td>
<td>4916</td>
</tr>
<tr>
<td>5-4</td>
<td>4948</td>
<td>4391</td>
</tr>
<tr>
<td>6-2</td>
<td>7079</td>
<td>6222</td>
</tr>
<tr>
<td>6-3</td>
<td>4665</td>
<td>4080</td>
</tr>
<tr>
<td>6-4</td>
<td>5925</td>
<td>5090</td>
</tr>
</tbody>
</table>

Table 31: The table showing the comparisons taking average response time when delays are changed by 10% and percentage improvement.

<table>
<thead>
<tr>
<th>QUERY_TYPE</th>
<th>50N(100 Queries)</th>
<th>75N(100 Queries)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AHY</td>
<td>HEURISTIC</td>
</tr>
<tr>
<td>3-2</td>
<td>7790</td>
<td>6927</td>
</tr>
<tr>
<td>3-3</td>
<td>6700</td>
<td>5829</td>
</tr>
<tr>
<td>3-4</td>
<td>5929</td>
<td>5205</td>
</tr>
<tr>
<td>4-2</td>
<td>5964</td>
<td>5299</td>
</tr>
<tr>
<td>4-3</td>
<td>6298</td>
<td>5613</td>
</tr>
<tr>
<td>4-4</td>
<td>4713</td>
<td>4188</td>
</tr>
<tr>
<td>5-2</td>
<td>8032</td>
<td>7051</td>
</tr>
<tr>
<td>5-3</td>
<td>8409</td>
<td>7423</td>
</tr>
<tr>
<td>5-4</td>
<td>7649</td>
<td>6742</td>
</tr>
<tr>
<td>6-2</td>
<td>7802</td>
<td>7006</td>
</tr>
<tr>
<td>6-3</td>
<td>8109</td>
<td>7240</td>
</tr>
<tr>
<td>6-4</td>
<td>5906</td>
<td>5217</td>
</tr>
</tbody>
</table>

Table 32: The table showing the comparisons taking average response time when delays are changed by 25% and percentage improvement.
BIBLIOGRAPHY


VITA AUCTORIS

Mohan Kumar was born in Hire Kagalur, Karnataka, India in 1970. He graduated from V.V.S. Sardar Patel High School — Bangalore, India in 1986. From there Mohan joined M.E.S. College of Arts, Commerce and Science — Bangalore for his Pre-University degree, which he obtained in 1988. Following this he pursued his career in Computer Science and Engineering by joining B.M.S. College of Engineering — Bangalore. He obtained his Bachelor’s degree in Engineering in 1992. Upon graduation, he joined P.E.S. Institute of Engineering and worked as an Assistant Systems Administrator for about six months. He is currently a candidate for a Master’s degree in Computer Science at the University of Windsor and will complete all degree requirements in the Fall of 1995. Mohan intends to work in the area of Computer Networks and Database Systems.