Establishing a Fuzzy Cost Model for Query Optimization in a Multidatabase System *

Qiang Zhu Per-Åke Larson

Department of Computer Science
University of Waterloo, Canada, N2L 3G1

Abstract

One of the challenges for query optimization in a multidatabase system (MDBS) is that some local optimization information may not be accurately known at the global level because of local autonomy. Traditional query optimization techniques using a crisp cost model may not be suitable for an MDBS because precise information is required. In this paper we present a new technique that performs query optimization using a fuzzy cost model that allows fuzzy information. We discuss methods for establishing a fuzzy cost model and introduce two fuzzy optimization criteria that can be used with a fuzzy cost model. We illustrate the benefits of such fuzzy query optimization. We also analyze the computational complexity for the fuzzy query optimization approach and suggest a simple method to reduce the complexity.

1 Introduction

A *multidatabase system* (MDBS) integrates data from pre-existing autonomous *local (component) databases* managed by heterogeneous database management systems (DBMS) in a distributed environment. It acts as a front end to multiple component DBMSs to provide full database functionality to global users, and it interacts with the component DBMSs at their external user interfaces. A key feature of an MDBS is the local autonomy that individual databases retain to serve existing applications[1]. Most differences between a conventional distributed database system (DDBS) and an MDBS are caused by local autonomy. These differences create some new challenges for query optimization in an MDBS[3, 4, 7]. Little work has been done, so far, on query optimization in an MDBS. Many issues remain unsolved.

Among the many challenges for query optimization in an MDBS, the crucial one is that some local optimization information, for example, local cost functions, local access methods and sizes of some tables, may not be known or accurately known by the global query optimizer because of local autonomy. How can we perform global query optimization in such a situation? This issue has recently been studied by several researchers. Du *et al.*[2] proposed an approach to deduce necessary local information by calibrating a given component DBMS. The idea is to construct a synthetic calibrating database and query it. Cost metrics for the queries are recorded and used to deduce the coefficients in the cost formula for the given component DBMS. Lu and Zhu *et al.*[3, 7, 8] suggested to perform probing/sampling queries on a (real) component database to obtain necessary optimization information. They also suggested performing adaptive (dynamic) query optimization based on runtime information in an MDBS.

In general, if an autonomous component DBMS is viewed as a black box whose optimization information is hidden from the global optimizer, there are three possible ways to obtain or estimate optimization information: (1) performing some test queries to probe the black box; (2) monitoring the behavior of the black box at runtime; and (3) guessing subjectively by using external characteristics of and previous knowledge about the black box. The first two ways have been studied in [2, 3, 4, 7, 8]. This paper explores a new approach with emphasis on the third way by exploiting fuzzy set theory.

All cost models for query optimization proposed so far are described by classical set theory. We argue that the problem of query optimization in an MDBS can be better described by using fuzzy set theory, introduced by Zadeh[6] in 1965, than by using classical (crisp) set theory. First of all, query optimization is a fuzzy

*Research supported by IBM Canada Laboratory and Natural Sciences and Engineering Research Council (NSERC) of Canada

notion (this is also true in traditional database systems). As we know, query optimization usually seeks a good or not-bad execution strategy for a query instead of a truly optimal strategy. What constitutes a good strategy, however, is vague. The class of good strategies has no well-defined boundary. This kind of fuzzy concept is exactly what the notion of a fuzzy set tries to capture. Second, as we said, some precise optimization information may not be available at the global level in an MDBS. We need to be tolerant of imprecise information. If a crisp mathematical model is used to evaluate the execution cost for a query, as in a centralized DBMS or a conventional DDBS, a suboptimal solution may result because imprecise information is used in an exact formula. A better way is to construct a fuzzy cost model that allows fuzzy coefficients and/or inputs for a formula. Third, an MDBS is usually more complicated than a conventional DDBS because it allows various types of heterogeneity and autonomy. It is difficult or sometimes impossible to give a precise description of the behavior and structure of an MDBS and its component DBMSs. A crisp cost model for such a system could be very complicated and inaccurate. Artificial and forced exactness may be used in the cost model. A fuzzy cost model simplifies the problem and accepts an imprecise description that is closer to people's actual perception of an MDBS. And last, fuzzy set theory provides us with a mechanism to build the subjective guesses of experts about optimization information into a fuzzy cost model, that is, subjectively construct the fuzzy sets employed in the fuzzy cost model.

The new query optimization approach for an MDBS in this paper is: (1) building a fuzzy cost model based on the knowledge, experience, tests and guesses of experts about the required optimization parameters, (2) improving the (fuzzy) cost model using runtime information collected by monitoring the system behavior, and (3) performing query optimization based on the fuzzy cost model to obtain a good execution strategy for a given query. Clearly, using a fuzzy cost model is the key point of this new approach. How to establish and use such a fuzzy cost model for an MDBS is the main topic of this paper.

The rest of this paper is organized as follows. Section 2 introduces basic concepts and notation used in this paper. Section 3 presents a fuzzy cost model for a particular MDBS and discusses the methods for establish such a fuzzy cost model. Section 4 discusses several issues for query optimization based on the fuzzy cost model. Section 5 gives an example to illustrate such fuzzy query optimization. The last section gives a summary and some future research issues.

2 Basic Concepts and Notation

Fuzzy set theory is a generalization of classical abstract set theory. Intuitively, a fuzzy set is a class that admits the possibility of partial membership in it. Let $X = \{x\}$ denote a space of objects (*universe*). Then a *fuzzy set* $\tilde{A}$ in X, denoted as $\tilde{A} \tilde{\subset} X$, is a set of ordered pairs

$$\tilde{A} = \{(x,\ \chi_{\tilde{A}}(x)) \mid x \in X\},$$

where $\chi_{\tilde{A}}$ is the *membership function* that maps X to a set L, called *membership space*, which is at least partially ordered. The value $\chi_{\tilde{A}}(x)$ is termed the *grade (degree) of membership* of the *element* x in $\tilde{A}$. Usually, the interval [0, 1] is taken as L, with 1 and 0 representing *full membership* and *nonmembership* in the fuzzy set $\tilde{A}$ respectively. If $\chi_{\tilde{A}}(x)$ is either 1 or 0 for all $x \in X$, $\tilde{A}$ becomes a classical set. The *support* of fuzzy set $\tilde{A}$ is a classical set

$$Supp(\tilde{A}) = \{x \mid \chi_{\tilde{A}}(x) \neq 0\}.$$

Unless otherwise stated, the membership space is assumed to be [0, 1] in this paper.

A fuzzy set $\tilde{A}$ is often also expressed as

$$\tilde{A} \ = \ \bigcup_{x \in X} \ \chi_{\tilde{A}}(x)/x \tag{1}$$

with $\chi_{\tilde{A}}(x)/x$ denoting the pair $(x\ , \chi_{\tilde{A}}(x))$. For $\bar{x} = \chi_{\tilde{A}}(x)/x \in \tilde{A}$, $grade(\bar{x})$ and $elem(\bar{x})$ denote $\chi_{\tilde{A}}(x)$ and x respectively. If X or $Supp(\tilde{A})$ is finite, then (1) can be written as

$$\tilde{A} = \{\chi_{\tilde{A}}(x_1)/x_1, \chi_{\tilde{A}}(x_2)/x_2, ..., \chi_{\tilde{A}}(x_n)/x_n\} \tag{2}$$

where $\chi_{\tilde{A}}(x_i) \neq 0$ for each $i = 1, ..., n$. In particular, if $\tilde{A} = \{1/a\}$, that is, a singleton crisp set, we simply write $\tilde{A} = a$ in this paper. If X is a subset of the real field $(-\infty, \infty)$, the *weighted average value* $\omega(\tilde{A})$ of the *fuzzy value* (set) $\tilde{A}$ of the form (2) is defined by the following formula:

$$\omega(\tilde{A}) = \frac{1}{W} \sum_{i=1}^{n} \chi_{\tilde{A}}(x_i) \ * \ x_i, \tag{3}$$

where

$$W = \sum_{j=1}^{n} \chi_{\tilde{A}}(x_j).$$

$\omega(\tilde{A})$ reflects the magnitude of the fuzzy value $\tilde{A}$ to some degree in a crisp way. In particular, if $\tilde{A} = a$, that is, a crisp value, $\omega(\tilde{A}) = a$.

There is a well-known extension principle[5] in fuzzy set theory that allows us to extend nonfuzzy functions, for example, arithmetic operations $(+, -, *)$, to a fuzzy environment. If a binary operation $\odot \in \{+,\ -,\ *\}$ on $x_1 \in X_1$ and $x_2 \in X_2$ with the result $y \in Y$, where $X_1,\ X_2$ and Y are subsets of $(-\infty,\ \infty)$, by the extension principle, a fuzzy binary operation $\tilde{\odot}$ on $\tilde{A}_1 \tilde{\subset} X_1$ and $\tilde{A}_2 \tilde{\subset} X_2$ can be defined as follows[1]:

$$\begin{aligned}\tilde{A}_1\ \tilde{\odot}\ \tilde{A}_2 = & \ (\bigcup_{x_1 \in X_1} \chi_{\tilde{A}_1}(x_1)/x_1) \\ & \qquad \tilde{\odot}\ (\bigcup_{x_2 \in X_2} \chi_{\tilde{A}_2}(x_2)/x_2) \\ = & \bigcup_{y \in \bar{Y}} \ [\ (\sup_{(x_1,x_2) \in X_1 \times X_2,\ y = x_1 \odot x_2} \\ & \quad min\{\chi_{\tilde{A}_1}(x_1), \chi_{\tilde{A}_2}(x_2)\})/y\ ], \end{aligned} \quad (4)$$

where $\bar{Y} = \{y \mid y = x_1 \odot x_2,\ (x_1, x_2) \in X_1 \times X_2\}$. Such a fuzzy operation allows us to carry the fuzzy information contained in its operands to its result. For example, if $\tilde{A}_1 = \{0.6/1,\ 1/2,\ 0.8/3\}$ and $\tilde{A}_2 = \{0.8/2,\ 1/3,\ 0.7/4\}$, then

$$\begin{aligned}\tilde{A}_1\ \tilde{*}\ \tilde{A}_2 \quad = \quad & \{0.6/2,\ 0.6/3,\ 0.8/4,\ 1/6,\ 0.7/8, \\ & 0.8/9,\ 0.7/12\}.\end{aligned}$$

where $\tilde{*}$ is the fuzzy multiplication extended from the arithmetic multiplication $*$ on real numbers. The first element $0.6/2$ in $\tilde{A}_1\ \tilde{*}\ \tilde{A}_2$ is obtained from $0.6/1$ in $\tilde{A}_1$ and $0.8/2$ in $\tilde{A}_2$ by the expression $min\{0.6,\ 0.8\}/(1 * 2)$. Since 2 results from $1 * 2$ that requires both 1 and 2, the possibility of 2 should be the possibility of the pair (1, 2), that is, the lower (min) possibility of 1 (in $\tilde{A}_1$) and 2 (in $\tilde{A}_2$). If there were more than one pair of values in $\tilde{A}_1$ and $\tilde{A}_2$ that produces 2, 2 could result from any of them. In that case, the possibility of 2 were the highest possibility of the alternative pairs. That is why *sup* is used in the definition of a fuzzy operation.

A fuzzy set captures the notion of inexactness, for example, vagueness or ambiguity. Theory developed for fuzzy sets has a broad application in solving problems that involve subjective evaluation. The assignment of the membership function of a fuzzy set is subjective in nature and, in general, reflects the context in which the problem is viewed.

[1] *sup* in (4) denotes the supremum, that is, the *least upper bound*. For a finite set, the supremum is the maximum.

3 Establishing a Fuzzy Cost Model

3.1 Assumptions

How can we use fuzzy set theory to establish a fuzzy cost model for query optimization in an MDBS? Let us illustrate this by outlining a fuzzy cost model for a particular MDBS. The following assumptions are made for the MDBS:

- There are N sites, and all sites are connected to each other.
- The common global data model is relational; that is, a relational interface is provided for each component DBMS in the MDBS although a component DBMS itself may not be relational.
- Join strategy, instead of semijoin strategy, is adopted.
- Joining tables must be on the same site in order for a join to be performed.
- An n-way join consists of a number of 2-way joins.

The cost of processing a global query in the MDBS consists of the data transmission cost and the local processing cost.

3.2 A fuzzy cost model

A cost function is used to estimate each type of costs. A set of cost functions together with their parameters and assumptions forms a cost model.

Performance information of a component DBMS or a network are usually reflected in the coefficients of cost functions. In an MDBS, however, such performance information may not be accurately known by the global query optimizer. In this case, fuzzy coefficients (sets) can be used in cost functions to allow imprecise information.

Let us consider an example. The startup cost of transmission from one site to another is usually assumed to be a constant. However, it is sometimes difficult to give such a constant precisely. An artificially precise value might lead to a bad execution strategy. In this case, a fuzzy constant (set) may be used to describe such a cost. For instance, using previous experience and some experiments, an expert may subjectively estimate the startup cost of transmission from site 1 to site 2 in the MDBS to be the following fuzzy constant:

$$\begin{aligned}\widetilde{C0} \quad = \quad & \{0.4/0.05,\ 0.8/0.08,\ 0.9/0.1, \\ & 0.75/0.12,\ 0.5/0.15\},\end{aligned}$$

which means that the degrees of possibility for the startup cost to be 0.05 sec., 0.08 sec., ... , and 0.15 sec. are 0.4, 0.8, ... , and 0.5 respectively. $\widetilde{C0}$ might be interpreted as the fuzzy value "approximately 0.1". Obviously, using a fuzzy constant in this situation is closer to people's actual observations in an MDBS because forced precision is avoided.

A cost function usually takes sizes of data as inputs. It is possible that exact size information about a local table may not be available at the global level in an MDBS. Also, estimating the size of intermediate results of subqueries for processing a query in an MDBS is much harder than estimating them in a conventional DDBS. These facts suggest that a cost function in an MDBS should also allow fuzzy inputs (sets). A cost function that allows fuzzy coefficients and/or fuzzy inputs is a fuzzy cost function[2]. A fuzzy cost function produces a fuzzy value that represents a "soft" estimate of the real cost, while a traditional crisp cost function produces a "hard" (crisp) estimate.

The fuzzy cost of transferring $\tilde{x}$ (fuzzy) bits of data from site i to site j is given by the following fuzzy function:

$$\tilde{\delta}_{ij}(\tilde{x}) \;=\; \widetilde{C0}_{ij} \;\tilde{+}\; \widetilde{C1}_{ij} \;\tilde{*}\; \tilde{x}, \tag{5}$$

where $\widetilde{C0}_{ij}$ is the fuzzy startup cost (set) for transmission and $\widetilde{C1}_{ij}$ is a fuzzy constant (set) that depends on channel bandwidth, error rate, distance and other line characteristics. $\tilde{\delta}_{ij}(\tilde{x}) = \tilde{\delta}_{ji}(\tilde{x})$ is assumed.

Most queries can be expressed as a sequence of select, project and join operations. For simplicity, we restrict ourselves to these three types of operations on a local site. The local processing cost consists of the costs for processing these three types of operations. Since a project operation is usually performed together with a select or join operation, no separate cost function will be given for it. Its cost will be reflected in the cost function for the relevant select or join operation.

For a select operation on a table R with a fuzzy selectivity (set) $\tilde{S}_\sigma$ on site i, its fuzzy cost is given by the following fuzzy function:

$$\begin{aligned}\tilde{\varphi}_i(\widetilde{|R|},\ \tilde{S}_\sigma) \;=\;& \widetilde{D0}_i \;\tilde{+}\; \widetilde{D1}_i \;\tilde{*}\; \widetilde{|R|} \\ &\tilde{+}\; \widetilde{D2}_i \;\tilde{*}\; \tilde{S}_\sigma \;\tilde{*}\; \widetilde{|R|},\end{aligned} \tag{6}$$

where the fuzzy input $\widetilde{|R|}$ is the fuzzy size (set) of the table R. The fuzzy coefficients $\widetilde{D0}_i$, $\widetilde{D1}_i$, and $\widetilde{D2}_i$ depend on the access method (for example, sequential scan, cluster-index-based scan), transformation due to heterogeneity, site processing speed, etc. The fuzzy size of the result table from this select operation is estimated as $\widetilde{|R|} * \tilde{S}_\sigma$.

In a traditional crisp cost model, only one probability distribution (for example, uniform distribution) is usually assumed about data within the attributes of a table. Different distributions may give different estimates for selectivities. Fuzzy selectivities allow us to assume multiple distributions, with different possibilities, about data in a table. For example, $\tilde{S}_\sigma = \{0.7/0.1, 0.4/0.3\}$ may indicate that the possibilities for the selectivity to be 0.1 (under uniform distribution) and 0.3 (under normal distribution) are 0.7 and 0.4 respectively. The traditional formulas to estimate selectivities under uniform distribution and normal distribution can be employed to compute the elements of the fuzzy selectivity.

For a join operation on two tables R_1 and R_2 with a fuzzy selectivity $\tilde{S}_\bowtie$ on site i, its fuzzy cost is given by the following fuzzy function:

$$\begin{aligned}\tilde{\eta}_i(\widetilde{|R_1|},\ \widetilde{|R_2|},\ \tilde{S}_\bowtie) \;=\;& \widetilde{E0}_i \;\tilde{+}\; \widetilde{E1}_i \;\tilde{*}\; \widetilde{|R_1|} \\ &\tilde{+}\; \widetilde{E2}_i \;\tilde{*}\; \widetilde{|R_2|} \;\tilde{+}\; \widetilde{E3}_i \;\tilde{*}\; \widetilde{|R_1|} \;\tilde{*}\; \widetilde{|R_2|} \\ &\tilde{+}\; \widetilde{E4}_i \;\tilde{*}\; \tilde{S}_\bowtie \;\tilde{*}\; \widetilde{|R_1|} \;\tilde{*}\; \widetilde{|R_2|},\end{aligned} \tag{7}$$

where the fuzzy coefficients $\widetilde{E0}_i$, $\widetilde{E1}_i$, $\widetilde{E2}_i$, $\widetilde{E3}_i$ and $\widetilde{E4}_i$ depend on the join algorithm (for example, nested-loop-join, merge-join, hash-join), transformation due to heterogeneity, site processing speed and so on. The fuzzy size of the result table from this join operation is estimated as $\widetilde{|R_1|} \;\tilde{*}\; \widetilde{|R_2|} \;\tilde{*}\; \tilde{S}_\bowtie$.

3.3 Methods for determining fuzzy parameters

The fuzzy parameters (inputs and coefficients) for the fuzzy functions from (5) $\sim$ (7) can be given or derived from experts' subjective estimates. The experts can be the developers of the global query optimizer, software and network administrators, experienced users and others[3]. There are three ways for the experts to make good fuzzy estimates:

- *Constructing a fuzzy parameter on the basis of some experiments.*

Testing queries can be performed to help experts to derive a fuzzy parameter for a fuzzy cost function.

[2]In fact, since a crisp set is special case of a fuzzy set, a crisp cost function may also be considered as a fuzzy cost function.

[3]Some experts, like local DBAs and developers of component DBMSs, may not be willing to divulge some information because of local autonomy and/or commercial reasons.

For example, after performing 20 testing select operations on some tables in a component database at site i, we find that 4 of them almost satisfy the following relationship

$$cost \;\approx\; 2.3 \;+\; 0.02 \;*\; |R| \;+\; 0.003 \;*\; S_\sigma \;*\; |R|$$

where $|R|$ is the cardinality of the table R in the component database and S_σ is the selectivity for the testing select operation, another 10 of the testing select operations almost satisfy

$$cost \;\approx\; 5.8 \;+ 0.1 \;*\; |R| \;+\; 0.09 \;*\; S_\sigma \;*\; |R|$$

and the remaining 6 of the testing select operations almost satisfy

$$cost \;\approx\; 4.1 \;+ 0.07 \;*\; |R| \;+\; 0.02 \;*\; S_\sigma \;*\; |R|,$$

Then we can specify the fuzzy function (6) at site i as follows

$$\begin{aligned}\tilde{\varphi}_i(\widetilde{|R|},\; \tilde{S}_\sigma) \;=\; & \{0.2/2.3,\; 0.5/5.8,\; 0.3/4.1\} \\ & \tilde{+}\;\; \{0.2/0.02,\; 0.5/0.1,\; 0.3/0.07\} \;\tilde{*}\; \widetilde{|R|} \\ & \tilde{+}\;\; \{0.2/0.003,\; 0.5/0.09,\; 0.3/0.02\} \;\tilde{*}\; \tilde{S}_\sigma \;\tilde{*}\; \widetilde{|R|}.\end{aligned}$$

If we know the fuzzy coefficients and the fuzzy input $\tilde{S}_\sigma$ of the fuzzy function (6) and we want to estimate the fuzzy size $\widetilde{|R|}$ of a table R at site i, we can perform testing queries on R to derive it. For example, we perform n testing queries on R and measure their execution time. These n testing queries are divided into k $(\leq n)$ groups $G_1, \ldots, G_k$. The testing queries in each group have a closer execution time than do the others. Let v_m $(1 \leq m \leq k)$ be the average execution time of the testing queries in the group G_m, and n_m be the number of testing queries in the group G_m. Observing (6), let r_m satisfy

$$v_m \;=\; \omega(\widetilde{D0}_i) + \omega(\widetilde{D1}_i) * r_m + \omega(\widetilde{D2}_i \;\tilde{*}\; \tilde{S}_\sigma) * r_m,$$

that is,

$$r_m \;=\; \frac{v_m \;-\; \omega(\widetilde{D0}_i)}{\omega(\widetilde{D1}_i) \;+\; \omega(\widetilde{D2}_i \;\tilde{*}\; \tilde{S}_\sigma)} \quad .$$

Then the fuzzy size $\widetilde{|R|}$ can be estimated as

$$\widetilde{|R|} \;=\; \{\frac{n_1}{n}/r_1,\; \frac{n_2}{n}/r_2,\; \ldots,\; \frac{n_k}{n}/r_k\}.$$

Many other types of experiments are possible.

- *Constructing a fuzzy parameter on the basis of the external characteristics of the object to be modeled.*

Although some internal information about an object, for example, component database, component DBMS and network, may not be known, a fuzzy parameter can be constructed by using the external characteristics of the object, for example, capability, processing speed, which may be obtained from the user documentation and experts' knowledge and experience. For example, assume site i and site j use the same type of DBMS, but the processing speed of site i is 10 times faster than that of site j, and the fuzzy cost function φ_i in (6) at site i is known, then the experts might estimate the cost for processing a select operation at site j to be approximately 10 times of that at site i. One possible fuzzy cost function φ_j at site j is

$$\begin{aligned}\varphi_j(\widetilde{|R|},\; \tilde{S}_\sigma) \;=\; & \{0.5/8,\; 0.9/10,\; 0.6/9\} \;\tilde{*}\; \widetilde{D0}_i \\ & \tilde{+}\;\; (\{0.4/5,\; 0.8/10,\; 0.7/8\} \;\tilde{*}\; \widetilde{D1}_i) \;\tilde{*}\; \widetilde{|R|} \\ & \tilde{+}\;\; (\{0.2/4,\; 0.8/10,\; 0.6/6\} \;\tilde{*}\; \widetilde{D2}_i) \;\tilde{*}\; \tilde{S}_\sigma \;\tilde{*}\; \widetilde{|R|}\end{aligned}$$

where $\{0.5/8,\ 0.9/10,\ 0.6/9\}$, $\{0.4/5,\ 0.8/10,\ 0.7/8\}$ and $\{0.3/7,\ 0.8/10,\ 0.6/6\}$ are the fuzzy proportional constants (different "approximately 10"s) between $\widetilde{Dl}_j$ and $\widetilde{Dl}_i$ $(l = 0, 1, 2)$, which reflect the experts' beliefs about the relationship between the costs at the two sides. The experts have probably taken into consideration other factors, such as site loads, in determining the fuzzy proportional constants.

- *Improving a fuzzy parameter on the basis of runtime information.*

Experts may make errors in constructing fuzzy parameters for the fuzzy cost model because of limited knowledge and information. Such errors can be corrected by using runtime information for executing user queries. The execution of a query can be monitored, and its runtime information can be collected by the global query optimizer. The grades of membership for elements in a fuzzy parameter can be dynamically adjusted according to runtime information.

3.4 Relationship between fuzzy and crisp models

Some fuzzy parameters in the fuzzy cost functions may not be fuzzy in some environments (sites). In such cases, they are reduced to crisp parameters, for example, $\widetilde{E0}_i = 3.56$ (i.e. $\{1/3.56\}$). If all fuzzy parameters are crisp, the fuzzy cost model boils down to a conventional crisp cost model. Therefore, a fuzzy cost model is a generalization of a conventional crisp cost model. The parameters for a crisp cost model can be considered as experts' crisp choices. In an MDBS, however, such crisp choices are likely to fail, because some information about component DBMSs in the MDBS is fuzzy in the experts' perception. A fuzzy cost model allows the experts to describe their fuzzy perception, which appears closer to the real world than a forced crisp cost model.

In this paper, we assume that the supports of all the fuzzy parameters are finite; that is, the experts give only a finite number of guesses for each fuzzy set.

4 Query Optimization Based on Fuzzy Cost Model

4.1 Fuzzy optimization and its criteria

Establishing a fuzzy cost model for an MDBS is not our ultimate purpose. Our ultimate purpose is to make use of such a fuzzy cost model to perform global query optimization in an MDBS. Let us see how to perform query optimization based on a fuzzy cost model.

For a given query, there are usually a number of feasible execution strategies. Let $X = \{x_1, x_2, \ldots, x_n\}$ be the set of all feasible strategies for a given query Q. For each strategy $x_i (1 \leq i \leq n)$, a fuzzy cost $\tilde{c}_i$ can be calculated by using the fuzzy cost model in Section 3.2. Let $C = \{\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_n\}$. The fuzzy cost model actually induces a fuzzy function $\tilde{F}: X \rightarrow C$. The problem of fuzzy query optimization is to find a strategy $x^0 \in X$ with the "smallest" fuzzy cost; that is, find x^0 such that

$$\tilde{F}(x^0) \;=\; \underset{x \in X}{{}''min''} \;\; \tilde{F}(x), \tag{8}$$

where $''min''$ operator is not in the traditional sense because comparisons among fuzzy costs are required.

Unlike a crisp (traditional) min operator, the $''min''$ operator in (8) can be defined in several ways. Different definitions reflect different decisions about how to make use of the fuzzy costs to find a possibly optimal strategy. Let us discuss two reasonable definitions for the $''min''$ operator. They differ in the ways for ordering fuzzy costs.

Let $\tilde{c}'$ and $\tilde{c}''$ be two fuzzy costs. The first way to define the ordering $\tilde{c}' \leq \tilde{c}''$ is that, if the (arithmetic) average of the values with the highest grade of membership in $\tilde{c}'$ is less or equal to the average of the values with the highest grade of membership in $\tilde{c}''$, that is, $\tilde{c}' \leq \tilde{c}''$ if $\widehat{avg}(\tilde{c}') \leq \widehat{avg}(\tilde{c}'')$, where

$$\widehat{avg}(\tilde{c}) \;=\; \underset{\bar{c} \in \tilde{c},\; grade(\bar{c}) = M}{avg} \{\; elem(\bar{c}) \;\}, \tag{9}$$

here $M = max\{grade(\bar{x}) | \bar{x} \in \tilde{c}\}$ and avg is the arithmetic average. Using this ordering, we can define $\underset{x \in X}{{}''min''} \; \tilde{F}(x)$ in (8) as a fuzzy cost $\tilde{c}_0 = \tilde{F}(x^0)$ (may not be unique) such that

$$\widehat{avg}(\tilde{c}_0) \;\leq\; \widehat{avg}(\tilde{F}(y)) \;, \quad \forall y \in X. \tag{10}$$

(10) gives rise to one possible fuzzy optimization criterion.

Example 4.1 Let $\tilde{c}_1 = \{0.4/76.3,\ 0.6/62.8,\ 0.5/68.8\}$, $\tilde{c}_2 = \{0.8/78.1,\ 0.8/75.2,\ 0.5/34.8,\ 0.6/21.0\}$, $\tilde{c}_3 = \{0.7/58.9,\ 0.7/162.3,\ 0.7/127.3,\ 0.4/219.3, 0.3/394.1\}$ and $\tilde{c}_4 = \{0.9/542.0, 0.6/359.9\}$ be the fuzzy costs of the strategies x_1, x_2, x_3 and x_4 for a query, respectively. According to the above definition of $''min''$, x_1 is the "optimal" strategy because

$$\begin{aligned} \widehat{avg}(\tilde{c}_1) = 62.8 \;\; < \;\; \widehat{avg}(\tilde{c}_2) &= \frac{78.1 + 75.2}{2} = 76.7 \\ < \;\; \widehat{avg}(\tilde{c}_3) &= \frac{58.9 + 162.3 + 127.3}{3} = 116.2 \\ < \;\; \widehat{avg}(\tilde{c}_4) &= 542.0 \;\; . \end{aligned} \tag{11}$$

The definition of $''min''$ in (10) puts the whole weight on the elements with the highest grade of membership in a fuzzy cost. It is intuitively correct because the fuzzy cost has the highest possibility to be one of the elements with the highest grade of membership. However, this definition ignores the elements with lower grades of membership in the fuzzy cost.

To take into consideration the elements with lower grades of membership in a fuzzy cost, we give another definition of the ordering between two fuzzy costs by using the weighted average values of the fuzzy costs, that is, $\tilde{c}_1 \leq \tilde{c}_2$ if $\omega(\tilde{c}_1) \leq \omega(\tilde{c}_2)$ where $\tilde{c}_1$ and $\tilde{c}_2$ are two fuzzy costs, and $\omega(\;)$ is defined in (3). Using this definition of ordering, we can define $\underset{x \in X}{{}''min''} \; \tilde{F}(x)$ in (8) as a fuzzy cost $\tilde{c}_0 = \tilde{F}(x^0)$ (may not be unique) such that

$$\omega(\tilde{c}_0) \;\leq\; \omega(\tilde{F}(y)) \;, \quad \forall y \in X. \tag{12}$$

(12) yields another possible optimization criterion.

According to the definition of $''min''$ in (12), the strategy x_2 is "optimal" among x_1, x_2, x_3 and x_4 in Example 4.1, because

$$\begin{aligned} \omega(\tilde{c}_2) &= \frac{1}{2.7}[0.8 * 78.1 + 0.8 * 75.2 + 0.5 * 34.8 \\ &\qquad + 0.6 * 21.0] = 56.5 \\ < \;\; \omega(\tilde{c}_1) &= \frac{1}{1.5}[0.4 * 76.3 + 0.6 * 62.8 \\ &\qquad + 0.5 * 68.8] = 68.4 \end{aligned}$$

$$
\begin{aligned}
< \ \omega(\tilde{c}_3) &= \frac{1}{2.8}[0.7 * 58.9 + 0.7 * 162.3 \\
&\quad + 0.7 * 127.3 + 0.4 * 219 + 0.3 * 394.1] \\
&= 160.6 \\
< \ \omega(\tilde{c}_4) &= \frac{1}{1.5}[0.9 * 542.0 + 0.6 * 359.9] \\
&= 469.2 \ .
\end{aligned}
$$

In fact, the two fuzzy optimization criteria induce two fuzzy sets that represent the same fuzzy notion "good execution strategy", that is,

$$\tilde{G}_1 = \{ -\widehat{avg}(\tilde{c})/x \mid x \ is \ a \ feasible \ strategy \ with \ fuzzy \ cost \ \tilde{c} \},$$

and

$$\tilde{G}_2 = \{ -\omega(\tilde{c})/x \mid x \ is \ a \ feasible \ strategy \ with \ fuzzy \ cost \ \tilde{c} \}.$$

The membership spaces of $\tilde{G}_1$ and $\tilde{G}_2$ are subsets of $(-\infty, \ 0]$ instead of $[0, \ 1]$. For each feasible strategy, $\tilde{G}_1$ and $\tilde{G}_2$ assign their own assessments of goodness to the strategy according to different criteria that reflect different subjective guesses about the goodness of the strategy. Within each fuzzy set, the higher the degree of goodness, the better the corresponding strategy may be. For such a fuzzy set, the task of a query optimizer is to find a strategy with a degree of goodness as high as possible. As a result, a good strategy for a query is usually obtained by the optimizer. In most cases, both $\tilde{G}_1$ and $\tilde{G}_2$ can be used to find a good strategy, for example, x_1 and x_2 in Example 4.1. Sometimes $\tilde{G}_2$ may produce a better strategy because all possible values are considered in estimating the degree of goodness for a strategy.

Let us now consider why the fuzzy approach has a higher chance of being better than the conventional crisp approach in an MDBS environment. As we said, some information, say C, is fuzzy at the global level in an MDBS. If we are forced to make a crisp estimate for C, we would give a value α_1 that is of the highest possibility to be C from our guess. If we are allowed to use a fuzzy set to describe C, we can specify not only α_1 but also other values with different lower possibilities, that is, $C = \{\chi(\alpha_1)/\alpha_1, \ \chi(\alpha_2)/\alpha_2, \ \ldots, \ \chi(\alpha_n)/\alpha_n\}$ where $\chi(\alpha_1) \geq \chi(\alpha_i)$ $(1 < i \leq n)$. The following theorem shows that the cost derived by using crisp estimates for an execution strategy for a query is always contained in the fuzzy cost derived by using fuzzy estimates for the strategy:

Theorem 1 *Let α be a crisp cost for a strategy x. It is derived from the cost model in Section 3.2 by using a crisp value for each parameter. If every parameter in the cost model is given by a fuzzy value that includes the corresponding crisp value used to derive α as one of its members with the highest grade of membership, and let $\tilde{C}$ be the fuzzy cost (for x) derived by using the fuzzy values, then α is one of the elements with the highest grade of membership in $\tilde{C}$.*

PROOF(IDEA). Let $\tilde{A}_1 = \{d_1/a_1, \ \ldots \}$ and $\tilde{A}_2 = \{d_2/a_2, \ \ldots \}$ where d_1 and d_2 are the highest grade of the membership in the fuzzy values $\tilde{A}_1$ and $\tilde{A}_2$ respectively. From (4), we have

$$\tilde{A}_1 \ \tilde{\odot} \ \tilde{A}_2 = \{ \ min\{d_1, \ d_2\}/(a_1 \odot a_2), \ \ldots \}.$$

where $\odot \in \{+, \ -, \ *\}$. Notice that $min\{d_1, \ d_2\}$ is the highest grade of membership in $\tilde{A}_1 \ \tilde{\odot} \ \tilde{A}_2$. From the assumptions of the theorem and the cost model in Section 3.2, we can see that the assertion of the theorem is true. ∎

If we use crisp estimates to derive the costs for the strategies x_1, x_2, x_3 and x_4 in Example 4.1, we may get the crisp estimate costs $\tilde{c}_1 = 62.8$, $\tilde{c}_2 = 78.1$, $\tilde{c}_3 = 58.9$ and $\tilde{c}_4 = 542.0$ by Theorem 1 (a crisp estimate cost is ensured to be in the corresponding fuzzy estimate cost with the highest grade of membership). From the costs, we would conclude that x_3 is an optimal strategy that is, in fact, most likely worse than the strategies x_1 and x_2 by observing the fuzzy costs given in Example 4.1. If we use the fuzzy costs given in Example 4.1, we can obtain the good strategy x_1 by using the first definition of $''min''$ or x_2 by using the second definition of $''min''$. The errors made from crisp choices may be adjusted by other values in a fuzzy set. That is the reason why a fuzzy cost model has a higher chance of being successful than a crisp one, especially in an MDBS environment where a correct crisp choice is usually hard to obtain.

4.2 Reducing computational complexity

Although fuzzy query optimization can usually find a good execution strategy, a problem with it is that its computational complexity may be much higher than conventional crisp query optimization. We have the following theorem:

Theorem 2 *Given an execution strategy x for a query, if using crisp values in the cost model in Section 3.2 to calculate the (crisp) cost for x requires n arithmetic operations, then using fuzzy values in the cost model to calculate the fuzzy cost $\tilde{C}$ for x requires*

$O(b^{n+1})$ (crisp) arithmetic operations where

$$b = max\{\ |Supp(\tilde{c})|\ \mid \tilde{c}\ is\ a\ fuzzy\ parameter\ used\ to\ calculate\ \tilde{C}\ \}$$

and $|Supp(\tilde{c})|$ is the cardinality of $Supp(\tilde{c})$. Assume $b > 1$.

PROOF(IDEA). Follow on from mathematical induction on n and (4). ∎

In practice, the complexity of the fuzzy approach may be lower if most of the fuzzy parameters are actually crisp values or many fuzzy operations have fewer elements in their results because of removals of duplicated elements during computation. However, in general, the computational complexity of the fuzzy approach can be as high as b^{n+1}. In other words, the complexity may grow exponentially as n increases. How can we solve this problem?

Notice that the elements with higher grades of membership in a fuzzy set are more possible to belong to the set than those with lower grades of membership in the fuzzy set. That is the reason why smaller weights are given to the elements with lower grades of membership when the weighted average value is calculated for a fuzzy value (set). Based on this property, a fuzzy value in the fuzzy cost model can be approximated by another fuzzy value that is obtained by removing some elements with low grades of membership, that is, set their grades of membership to zeros.

Let $\tilde{C} = \{d_1/c_1,\ \dots\ ,\ d_k/c_k,\ d_{k+1}/c_{k+1},\ \dots\ \}$ be a fuzzy value where $d_1 \geq \dots \geq d_k \geq d_{k+1} \geq \dots$. A *truncated k-approximate fuzzy value* for $\tilde{C}$ is defined as $\tilde{C}^{(k)} = \{d_1/c_1,\ \dots\ ,\ d_k/c_k\}$. If $k > |Supp(\tilde{C})|$, assume $\tilde{C}^{(k)} = \tilde{C}$. A truncated k-approximate fuzzy value for a given fuzzy value may not be unique, for example, both $\tilde{A}^{(2)'} = \{0.8/23.1,\ 0.6/65.2\}$ and $\tilde{A}^{(2)''} = \{0.8/23.1,\ 0.6/48.3\}$ are truncated 2-approximate fuzzy values for the fuzzy value $\tilde{A} = \{0.6/65.2,\ 0.6/48.3,\ 0.6/50.4,\ 0.8/23.1,\ 0.5/75.3,\ 0.3/21.9\}$.

Lemma 1 *Let $\tilde{A}_1^{(k)}$ and $\tilde{A}_2^{(k)}$ be two truncated k-approximate fuzzy values for the fuzzy values $\tilde{A}_1$ and $\tilde{A}_2$ respectively. Then $\tilde{A}_1^{(m)}\ \tilde{\odot}\ \tilde{A}_2^{(m)}$ is a truncated k-approximate fuzzy value for $\tilde{A}_1\ \tilde{\odot}\ \tilde{A}_2$, where $\odot \in \{+,\ -,\ *\}$.*

PROOF(IDEA). By definition and (4). ∎

Lemma 1 says that truncated k-approximate fuzzy operands can be used to derive a truncated k-approximate fuzzy result for a fuzzy arithmetic operation. Applying Lemma 1 repeatedly to the cost model in Section 3.2 to calculate a truncated k-approximate fuzzy cost (value) of a strategy for a query will reduce the computational complexity greatly. We have the following theorem:

Theorem 3 *Given an execution strategy x for a query, if using crisp values in the cost model of Section 3.2 to calculate the (crisp) cost for x requires n (crisp) arithmetic operations, then using truncated k-approximate fuzzy values to approximate all the fuzzy parameters and intermediate results in the cost model to calculate a truncated k-approximate fuzzy cost for the fuzzy cost for x requires $O(k^2 * n)$ (crisp) arithmetic operations.*

PROOF(IDEA). Notice that every fuzzy binary operation with two truncated k-approximate operands requires at most k^2 (crisp) arithmetic operations. ∎

If $k = 1$, the fuzzy approach is reduced to a crisp approach. By choosing an appropriate $k > 1$, we can get an efficient fuzzy optimization method. One of the reasonable choices is setting

$$k = max\{\ |Supp(\tilde{c})|\ \mid \tilde{c}\ is\ a\ fuzzy\ parameter\ in\ the\ cost\ model\}.$$

Since such k is a finite number in our cost model, the complexity of the fuzzy optimization method in Theorem 3 is of the same order of the complexity as the corresponding crisp optimization method.

5 An Example

Let R_1 and R_2 be two tables at site 1 and site 2, respectively. For the distributed join $R_1 \bowtie R_2$, there are the following two execution strategies (among many others):

s_1: transfer R_1 to site 2 and perform the join at site 2.

s_2: transfer R_2 to site 1 and perform the join at site 1.

Assume that the local processing costs at both sites can be neglected, comparing with the communication costs, that is, $\widetilde{Ej}_i = 0$ $(0 \leq j \leq 4; i = 1, 2)$ in the cost function (7). Hence the costs for executing s_1 and s_2 are determined by the communication costs that are computed by using the cost function (5). We want to choose a cheaper strategy between s_1 and s_2. The real values, estimated fuzzy values and estimated crisp values about the coefficients and inputs in (5) are given in Table 1. The real values may not be known by the

TABLE 1: Values for Parameters tl_1 — length of tuple in R_1, tl_2 — length of tuple in R_2			
	real value	estimated fuzzy value	estimated crisp value
$\widetilde{C0}_{12}$	2.8	{0.8/2.8, 0.3/5.5, 0.2/3.5}	2.8
$\widetilde{C1}_{12}$	0.0004	{0.6/0.0002, 0.8/0.0008, 0.5/0.0001}	0.0008
$\widetilde{\lvert R_1\rvert}$	1100	{0.8/1200, 0.4/980}	1200
tl_1	504	504	504
$\widetilde{\lvert R_2\rvert}$	1600	{0.9/1100, 0.7/1700}	1100
tl_2	480	480	480
$\tilde{x}_1 = \widetilde{\lvert R_1\rvert}\tilde{*}tl_1$	554400	{0.8/604800, 0.4/493920}	604800
$\tilde{x}_2 = \widetilde{\lvert R_2\rvert}\tilde{*}tl_2$	768000	{0.9/528000, 0.7/816000}	528000

global query optimizer. They are listed here for the purpose of comparison.

Using real values in Table 1, we have

$$\tilde{\delta}_{12}(\tilde{x}_1) = 2.8 + 0.0004 * 554400 = 224.6 ,$$

and

$$\tilde{\delta}_{21}(\tilde{x}_2) = 2.8 + 0.0004 * 864000 = 310.0 ,$$

which are the real costs for executing s_1 and s_2, respectively. Since $\tilde{\delta}_{12}(\tilde{x}_1) < \tilde{\delta}_{12}(\tilde{x}_2)$, the strategy s_1 is more efficient than s_2.

Using fuzzy values in Table 1, we have

$$\begin{aligned}\tilde{\delta}_{12}(\tilde{x}_1) = {} & \{0.8/2.8,\ 0.3/5.5,\ 0.2/3.5\} \tilde{+} \\ & \{0.6/0.0002,\ 0.8/0.0008,\ 0.5/0.0001\} \\ & \tilde{*} \{0.8/604800,\ 0.4/493920\} \\ = {} & \{0.6/123.8,\ 0.4/101.6,\ 0.8/486.6,\ 0.4/397.9, \\ & 0.5/63.3,\ 0.4/52.2,\ 0.3/126.5,\ 0.3/104.3, \\ & 0.3/489.3,\ 0.3/400.6,\ 0.3/66.0,\ 0.3/54.9, \\ & 0.2/124.5,\ 0.2/102.3,\ 0.2/487.3,\ 0.2/398.6, \\ & 0.2/64.0,\ 0.2/52.9\},\end{aligned}$$

and

$$\begin{aligned}\tilde{\delta}_{21}(\tilde{x}_2) = {} & \{0.8/2.8,\ 0.3/5.5,\ 0.2/3.5\} \tilde{+} \\ & \{0.6/0.0002,\ 0.8/0.0008,\ 0.5/0.0001\} \\ & \tilde{*} \{0.9/528000,\ 0.7/816000\} \\ = {} & \{0.6/108.4,\ 0.6/166.0,\ 0.8/425.2,\ 0.7/655.6, \\ & 0.5/55.6,\ 0.5/84.4,\ 0.3/111.1,\ 0.3/168.7, \\ & 0.3/427.9,\ 0.3/658.3,\ 0.3/58.3,\ 0.3/87.1, \\ & 0.2/109.1,\ 0.2/166.7,\ 0.2/425.9,\ 0.2/656.3, \\ & 0.2/56.3,\ 0.2/85.1\}.\end{aligned}$$

By the definition of $''min''$ in (12), we choose s_1 as a better strategy because $\omega(\tilde{\delta}_{12}(\tilde{x}_1)) = 218.7 < \omega(\tilde{\delta}_{21}(\tilde{x}_2)) = 266.7$.

Using truncated 3-approximate fuzzy values to calculate the costs, we have

$$\begin{aligned}\tilde{\delta}_{12}(\tilde{x}_1)^{(3)} = {} & [\{0.8/2.8,\ 0.3/5.5,\ 0.2/3.5\} \tilde{+} \\ & (\{0.6/0.0002,\ 0.8/0.0008,\ 0.5/0.0001\} \\ & \tilde{*} \{0.8/604800,\ 0.4/493920\})^{(3)}]^{(3)} \\ = {} & [\{0.8/2.8,\ 0.3/5.5,\ 0.2/3.5\} \\ & \tilde{+} \{0.6/121.0,\ 0.8/483.8,\ 0.5/60.5\}]^{(3)} \\ = {} & \{0.6/123.8,\ 0.8/486.6,\ 0.5/63.3\},\end{aligned}$$

and

$$\begin{aligned}\tilde{\delta}_{21}(\tilde{x}_2)^{(3)} = {} & [\{0.8/2.8,\ 0.3/5.5,\ 0.2/3.5\} \tilde{+} \\ & (\{0.6/0.0002,\ 0.8/0.0008,\ 0.5/0.0001\} \\ & \tilde{*} \{0.9/528000,\ 0.7/816000\})^{(3)}]^{(3)} \\ = {} & [\{0.8/2.8,\ 0.3/5.5,\ 0.2/3.5\} \\ & \tilde{+} \{0.6/105.6,\ 0.8/422.4,\ 0.7/652.8\}]^{(3)} \\ = {} & \{0.6/108.4,\ 0.8/425.2,\ 0.7/655.6\}.\end{aligned}$$

Since $\omega(\tilde{\delta}_{12}(\tilde{x}_1)^{(3)}) = 260.6 < \omega(\tilde{\delta}_{21}(\tilde{x}_2)^{(3)}) = 411.5$, s_1 is chosen as a better strategy again by (12). Clearly, using truncated 3-approximate fuzzy values can reduce a lot of computations and still keep most information in the original fuzzy values.

Using the estimated crisp values in Table 1, we have

$$\begin{aligned}\tilde{\delta}_{12}(\tilde{x}_1) &= 2.8 + 0.0008 * 604800 = 486.6 \\ > \tilde{\delta}_{21}(\tilde{x}_2) &= 2.8 + 0.0008 * 528000 \\ &= 425.2 ,\end{aligned}$$

which gives a wrong conclusion that s_2 is better. The reason is that the crisp values used are not accurate. A fuzzy value reflects experts' spontaneous fuzzy perception of an MDBS. Errors made in a forced crisp value can be corrected by the values with lower grades of membership in the fuzzy value. Hence a fuzzy approach usually can make a correct decision in choosing a good execution strategy for a query in an MDBS.

6 Conclusion

How to perform query optimization when some precise optimization information is not available at the global level in an MDBS is a new challenge. This paper presents a new approach using fuzzy set theory to tackle this challenge. The idea is to build a fuzzy cost model on the basis of experts' actual fuzzy perceptions of an MDBS environment and perform query optimization using the fuzzy cost model to choose a good execution strategy for a given query. It is shown that, in an MDBS environment, the fuzzy query optimization approach is usually better in choosing a good execution strategy than a traditional query optimization approach that uses a crisp cost model.

A fuzzy cost model for an MDBS is established to demonstrate how fuzzy information can be used to estimate the cost of an execution strategy for a query. The fuzzy parameters in a fuzzy cost model can be determined and improved by experiments, external characteristics of objects and runtime information. It is shown that a fuzzy estimated cost usually contains more information than a crisp estimated cost. Two reasonable criteria are suggested for query optimization using a fuzzy cost model. However, the computational complexity of a raw fuzzy query optimization method may be higher than that of the relevant crisp query optimization method. An efficient fuzzy query optimization method using truncated k-approximate fuzzy values is, therefore, suggested. This fuzzy query optimization method is proven to have the same order of complexity as the relevant crisp query optimization method.

We plan to further explore other methods to establish a good fuzzy cost model for an MDBS and investigate fuzzy optimization algorithms, for example, fuzzy linear programming, fuzzy dynamic programming, based on a fuzzy cost model. We also plan to develop some heuristics based on fuzzy information to reduce the complexity of fuzzy query optimization.

References

[1] M. W. Bright, A. R. Hurson, and S. H. Pakzad. A taxonomy and current issues in multidatabase systems. *IEEE Computer*, pages 50–59, March 1992.

[2] W. Du, R. Krishnamurthy, and M. C. Shan. Query optimization in heterogeneous DBMS. In *Proceedings of VLDB*, pages 277–91, 1992.

[3] H. Lu, B.-C. Ooi, and C.-H. Goh. On global multidatabase query optimization. *SIGMOD Record*, 21(4):6–11, Dec. 1992.

[4] H. Lu and M.-C. Shan. On global query optimization in multidatabase systems. In *2nd Int'l workshop on Research Issues on Data Eng.*, page 217, Tempe, Arizona, USA, 1992.

[5] V. Novak. *Fuzzy Sets and Their Applications.* Adam Hilger, 1989.

[6] L. A. Zadel. Fuzzy sets. *Inform. and Control*, 8:338–53, 1965.

[7] Qiang Zhu. Query optimization in multidatabase systems. In *Proceedings of the 1992 CAS Conference, vol.II*, pages 111–27, Toronto, Canada, Nov. 1992.

[8] Qiang Zhu and P.-Å. Larson. A query sampling method of estimating local cost parameters in a multidatabase system. In *Proceedings of the 10th International Conference on Data Engineering (to appear)*, Houston, Texas, Feb. 1994.