International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems
Vol. 5, No. 6 (1997) 701—722

A FUZZY QUERY OPTIMIZATION APPROACH FOR MULTIDATABASE SYSTEMS

QIANG ZHU
Department of Computer and Information Science
University of Michigan
Dearborn, MI 48128, USA

PER-ÅKE LARSON*
Department of Computer Science
University of Waterloo
Ontario, Canada, N2L 3G1

Received (February 1997)
Revised (August 1997)

A crucial challenge for global query optimization in a multidatabase system (MDBS) is that some local optimization information, such as local cost parameters, 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 approach that performs global query optimization using a fuzzy cost model that allows fuzzy information. We suggest methods for establishing a fuzzy cost model and introduce a fuzzy optimization criterion that can be used with a fuzzy cost model. We discuss the relationship between the fuzzy optimization approach and the traditional (crisp) optimization approach and show that the former has a better chance to find a good execution strategy for a query in an MDBS environment, but its complexity may grow exponentially compared with the complexity of the later. To reduce the complexity, we suggest to use so-called k-approximate fuzzy values to approximate all fuzzy values during fuzzy query optimization. It is proven that the improved fuzzy approach has the same order of complexity as the crisp approach.

Keywords: multidatabase system, global query optimization, fuzzy cost model, fuzzy optimization, approximate fuzzy value.

1. Introduction

A multidatabase system (MDBS) integrates data from pre-existing autonomous local databases managed by heterogeneous local database management systems (DBMS) in a distributed environment. It acts as a front end to multiple local DBMSs to provide full database functionality for global users, and it interacts with

*Current address: Microsoft Corporation, One Microsoft Way, Redmond, WA 98052–6399, USA.

the local 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] Major differences between a conventional distributed database system (DDBS) and an MDBS are caused by local autonomy. These differences raise some new challenges for global query optimization in an MDBS.[2,3,4,5,6] Little work has been done, so far, on global query optimization in an MDBS. Many issues remain unsolved.

Among many challenges for global query optimization in an MDBS, the crucial one is that some local optimization information, such as local cost parameters, local access methods, and some local table sizes, 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.*[7] proposed an approach to deduce necessary local cost parameters by calibrating a given local 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 local DBMS. Zhu and Larson[8,9] suggested to perform sampling queries on a real local database to obtain necessary optimization information. Lu and Zhu[2,4] discussed issues for performing adaptive (dynamic) query optimization based on runtime information in an MDBS.

In general, if an autonomous local DBMS is viewed as a black box whose optimization information is hidden from the global query optimizer, there are three possible ways to obtain or estimate optimization information (see Fig. 1): (1) performing some test queries to probe the black box; (2) monitoring the behavior of

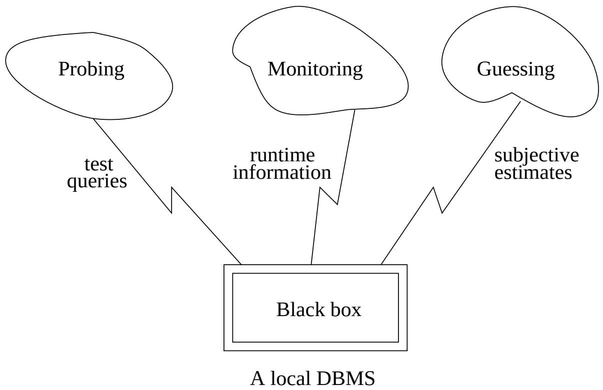

Fig. 1. Ways to derive local information

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 the literature.[2,3,4,8,7,9,10] 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 based on classical set theory. However, the problem of (global) query optimization in an MDBS can be better described by using fuzzy set theory, introduced by Zadeh[11] in 1965, than by using classical (crisp) set theory. There are several reasons for this:

- Query optimization is a fuzzy notion. 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 can capture.

- 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 bad 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.

- A fuzzy cost model simplifies the description of an MDBS. 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 structures of an MDBS and its local 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.

- Fuzzy set theory provides us with a mechanism to build experts' subjective guesses about local optimization information into a fuzzy cost model, that is, to subjectively construct the fuzzy parameters (sets) employed in the fuzzy cost model. Although precise local optimization information may not be available in an MDBS, some fuzzy information, such as "local DBMS A is approximately twice as fast as local DBMS B", may be useful.

The idea of the fuzzy query optimization approach for an MDBS proposed in this paper is: (1) building a fuzzy cost model based on experts' knowledge, experience, tests and guesses about the required optimization parameters; and (2) performing query optimization based on the fuzzy cost model to obtain a good execution strategy for a given query. The relevant issues on fuzzy query optimization are to be discussed in this paper.

Although fuzzy set theory has been applied to the database area for many years,[12] it was mainly used to represent imprecise data in databases (i.e., fuzzy databases)[13,14,15,16,17] and develop fuzzy queries to retrieve imprecise data.[18,19,20,21] A number of related issues, such as functional dependencies, security, implementation considerations and others, have also been investigated.[22,23,24] This paper is to discuss how to use fuzzy information to process/optimize crisp queries (instead of fuzzy queries) on crisp databases (instead of fuzzy databases). To our knowledge, no similar work has been done by other researchers in the literature.

The rest of the paper is organized as follows. Section 2 introduces basic concepts and notation used in this paper. Section 3 presents a fuzzy cost model for an MDBS and discusses the methods for establishing such a fuzzy cost model. Section 4 defines the problem of fuzzy query optimization and discusses the relationship between the fuzzy and crisp approaches. Section 5 discusses how to reduce the complexity of the fuzzy optimization approach. The last section gives a summary and some future research issues.

2. Basic Concepts

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.

Definition 1 *Let $X = \{x\}$ denote a space of objects (universe). 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. $\chi_{\tilde{A}}(x)$ is termed the grade (degree) of membership of element x in $\tilde{A}$.

Usually, the interval [0, 1] is taken as L. If $\chi_{\tilde{A}}(x)$ is either 1 or 0 for every $x \in X$, $\tilde{A}$ becomes a classical set. Unless otherwise stated, the membership space is assumed to be [0, 1] in this paper.

Definition 2 *The support of fuzzy set $\tilde{A}$ is a classical set*

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

If $Supp(\tilde{A})$ is finite, $\tilde{A}$ is called a finite fuzzy set.

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 $\bar{x} = \chi_{\tilde{A}}(x)/x \in \tilde{A}$, let $grade(\bar{x})$ and $elem(\bar{x})$ denote $\chi_{\tilde{A}}(x)$ and x, respectively. For a finite fuzzy set, (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.

In addition to the standard concepts of fuzzy sets in Definitions 1 and 2,[25] we introduce the following concept of a weighted average value for a finite fuzzy value (set).

Definition 3 *If X is a subset of the real field (-∞, ∞), $\tilde{A} \tilde{\subset} X$ is called a fuzzy value. For a finite fuzzy value $\tilde{A} = \{\chi_{\tilde{A}}(x_1)/x_1, \chi_{\tilde{A}}(x_2)/x_2, ..., \chi_{\tilde{A}}(x_n)/x_n\}$, the weighted average value $\omega(\tilde{A})$ of $\tilde{A}$ 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, $\omega(\tilde{A}) = a$.

A fuzzy set captures the notion of inexactness, such as 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.

To develop a fuzzy cost model for an MDBS, one needs to know how to perform arithmetic operations, such as addition and multiplication, on fuzzy values. The following extension principle[25] in fuzzy set theory provides a general extension of non-fuzzy functions, such as arithmetic operations, to a fuzzy environment.

Extension Principle *Let X be a Cartesian product of n universes $X = X_1 \times X_2 \times \cdots \times X_n$, and $\tilde{A}_1, \ \tilde{A}_2, \ \ldots \ , \ \tilde{A}_n$ be n fuzzy sets in $X_1, \ X_2, \ \ldots \ , \ X_n$, respectively. f is a function from X to a universe Y. Then a fuzzy set $\tilde{B}$ in Y can be induced as follows*

$$\tilde{B} = \{ \ (y, \chi_{\tilde{B}}(y)) \mid y = f(x_1, ..., x_n), \ (x_1, ..., x_n) \in X \ \}$$

where[1]

$$\chi_{\tilde{B}}(y) = \begin{cases} \sup_{(x_1,\ldots,x_n) \in f^{-1}(y)} min\{\chi_{\tilde{A}_1}(x_1), \ldots, \chi_{\tilde{A}_n}(x_n)\} & \\ \qquad\qquad if \ f^{-1}(y) \neq empty \ , & \\ 0 & otherwise \end{cases} \tag{4}$$

here f^{-1} is the inverse of f.

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

If f is a binary arithmetic operation $\odot \in \{+,\ -,\ *\}$ on $X_1 \times X_2$ with values in Y, where $X_1,\ X_2$ and Y are subsets of $(-\infty,\ \infty)$, the extension principle defines a fuzzy binary arithmetic operation $\tilde{\odot}$ as follows:

$$\begin{aligned} \tilde{A_1}\ \tilde{\odot}\ \tilde{A_2} &= (\bigcup_{x_1 \in X_1} \chi_{\tilde{A_1}}(x_1)/x_1) \quad \tilde{\odot} \quad (\bigcup_{x_2 \in X_2} \chi_{\tilde{A_2}}(x_2)/x_2) \\ &= \bigcup_{y \in \bar{Y}} \ [\ (\sup_{(x_1,x_2) \in X_1 \times X_2,\ y = x_1 \odot x_2} \quad min\{\chi_{\tilde{A_1}}(x_1), \chi_{\tilde{A_2}}(x_2)\})/y\ ]\ , \end{aligned}$$

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 $\tilde{A_1} = \{0.5/1,\ 0.9/2,\ 0.8/3\}$ and $\tilde{A_2} = \{0.2/2,\ 0.7/3,\ 0.6/4\}$, then

$$\tilde{A_1}\ \tilde{*}\ \tilde{A_2} = \{0.2/2,\ 0.5/3,\ 0.5/4,\ 0.7/6,\ 0.6/8,\ 0.7/9,\ 0.6/12\}\ .$$

where $\tilde{*}$ is the fuzzy multiplication extended from the arithmetic multiplication $*$ on real numbers. The first element $0.2/2$ in $\tilde{A_1}\ \tilde{*}\ \tilde{A_2}$ is obtained from $0.5/1$ in $\tilde{A_1}$ and $0.2/2$ in $\tilde{A_2}$ by expression $min\{0.5,\ 0.2\}/(1*2)$. Since 2 results from $1*2$ that requires both 1 and 2, the possibility of 2 should be the possibility of pair $(1,\ 2)$, that is, the lower 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.

3. Fuzzy Cost Model

How can we use fuzzy set theory to establish a fuzzy cost model for global query optimization in an MDBS? Let us illustrate the idea by outlining a fuzzy cost model for a particular MDBS.

3.1. *Assumptions*

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 local DBMS in the MDBS although a local 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 a data transmission cost and a local processing cost.

3.2. *A fuzzy cost model*

Each type of cost is estimated by a cost function. All cost functions together with their parameters and assumptions forms a cost model for the MDBS. Using the cost model, the global query optimizer can choose a good execution strategy for a query by comparing the costs of alternative strategies.

Performance information of a local DBMS or a network is 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 in an MDBS. However, it is sometimes difficult to determine such a constant precisely. An artificially precise value may mislead the global query optimizer into choosing 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 a particular MDBS to be the following fuzzy constant:

$$\widetilde{C0} \;=\; \{0.4/0.05,\ 0.7/0.08,\ 0.9/0.1,\ 0.8/0.12,\ 0.6/0.15\}\ ,$$

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, 0.7, ... , and 0.6 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 the sizes of tables in a database as inputs. It is possible that exact size information about a table at a local site may not be available at the global level in an MDBS. Also, estimating the sizes of intermediate result tables 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 can be estimated 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.

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

As we know, most common queries can be expressed as a sequence of select, project and join operations. For simplicity, we restrict ourselves to these three types of operations at 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 related select or join operation. A select operation that may or may not be followed by a project operation is called a unary query. A (2-way) join operation that may or may not be followed by a project operation is called a join query.

For a unary query that is performed by a scan access method k (such as sequential scan, indexed-based scan, or hash-based scan) on a table R with a fuzzy selectivity (set) $\tilde{S}_\sigma$ at site i, its fuzzy cost is estimated by the following fuzzy function:

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

where fuzzy input $\widetilde{|R|}$ is the fuzzy size (set) of table R. The fuzzy coefficients $\widetilde{D0}_{ik}$, $\widetilde{D1}_{ik}$, and $\widetilde{D2}_{ik}$ depend on the performance of scan access method k at site i. They can be interpreted as the fuzzy initialization cost, the fuzzy cost of retrieving a tuple from operand table R, and the fuzzy cost of processing a result tuple, respectively. The fuzzy size of the result table for this unary query is estimated as $\widetilde{|R|} * \tilde{S}_\sigma$.

In a traditional crisp cost model, only one (probability) distribution, such as uniform distribution, is usually assumed about data for the attributes of a table. However, different distributions may yield different estimates for selectivities. Fuzzy selectivities allow multiple distributions, with different possibilities, about data in a table. For example, $\tilde{S_\sigma} = \{1.0/0.1, 0.6/0.3\}$ may indicate that the relative possibilities for the selectivity to be 0.1 (under uniform distribution) and 0.3 (under normal distribution) are 1.0 and 0.6 respectively. The traditional formulas to estimate selectivities under uniform distribution and normal distribution can be employed to estimate the elements of the fuzzy selectivity.

For a join query that is performed by a join access method k (such as nested loop, index-based join, hash join, or sort-merge join) on two tables R_1 and R_2 with a fuzzy selectivity $\tilde{S}_\bowtie$ at site i, its fuzzy cost is estimated by the following fuzzy function:

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

where the fuzzy coefficients $\widetilde{E0}_{ik}$, $\widetilde{E1}_{ik}$, $\widetilde{E2}_{ik}$, $\widetilde{E3}_{ik}$ and $\widetilde{E4}_{ik}$ depend on the performance of join access method k at site i. They can be interpreted as the fuzzy initialization cost, the fuzzy cost of preprocessing a tuple in R_1, the fuzzy cost of preprocessing a tuple in R_2, the fuzzy cost of processing a tuple in the Cartesian product of the two operand tables, and the fuzzy cost of processing a result tuple, respectively. The fuzzy size of the result table from this join operation is estimated as $\widetilde{|R_1|} \tilde{*} \widetilde{|R_2|} \tilde{*} \tilde{S}_{\bowtie}$.

Applying the fuzzy cost functions in the fuzzy cost model, we can derive a fuzzy cost estimate of an execution strategy for a global query. For example, assume that query $R_1 \bowtie R_2 \bowtie R_3$ is issued at site 0 ($\bowtie$ denotes the natural join), where R_i is a table in the local database at site i ($i = 1, 2, 3$), the following is a feasible execution strategy to execute it:

- Transfer R_1 from site 1 to site 2 and employ the index-based join method (say, it is numbered as join access method 2) to perform $R_{12} = R_1 \bowtie R_2$ at site 2.
- Transfer R_{12} from site 2 to site 3 and employ the nested loop method (say, it is numbered as join access method 1) to perform $R_{123} = R_{12} \bowtie R_3$ at site 3.
- Finally, send R_{123} to site 0.

Let the fuzzy tuple lengths of R_1, R_{12} and R_{123} be $\widetilde{tl}_1$, $\widetilde{tl}_{12}$ and $\widetilde{tl}_{123}$, respectively. Let the fuzzy selectivities of two joins be $\tilde{S}_{\bowtie 12}$ and $\tilde{S}_{\bowtie 123}$, respectively. Then the total fuzzy cost of the execution strategy is estimated by the following expression

$$\tilde{\delta}_{12}(\widetilde{|R_1|} \tilde{*} \widetilde{tl}_1) \tilde{+} \tilde{\eta}_{22}(\widetilde{|R_1|},\ \widetilde{|R_2|},\ \tilde{S}_{\bowtie 12}) \tilde{+} \tilde{\delta}_{23}(\widetilde{|R_{12}|} \tilde{*} \widetilde{tl}_{12})$$
$$\tilde{+} \tilde{\eta}_{31}(\widetilde{|R_{12}|},\ \widetilde{|R_3|},\ \tilde{S}_{\bowtie 123}) \tilde{+} \tilde{\delta}_{30}(\widetilde{|R_{123}|} \tilde{*} \widetilde{tl}_{123})\ ,$$

where $\widetilde{|R_{12}|} = \widetilde{|R_1|} \tilde{*} \widetilde{|R_2|} \tilde{*} \tilde{S}_{\bowtie 12}$ and $\widetilde{|R_{123}|} = \widetilde{|R_{12}|} \tilde{*} \widetilde{|R_3|} \tilde{*} \tilde{S}_{\bowtie 123}$.

3.3. *Methods for determining fuzzy parameters*

The fuzzy parameters (inputs and coefficients) for the fuzzy functions in (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, which are described below.

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

Test queries can be performed to help experts to estimate the fuzzy parameters for a fuzzy cost function. For example, after performing 20 unary test queries that employ an index-based scan[4](say, it is numbered as scan access method 2) against a local database at site i, we find that four 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 table R in the local database and S_σ is the selectivity of the unary test query, another 10 of the unary test queries almost satisfy

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

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

[4]See the references[8,9] about how to classify local queries according to their access methods.

and the remaining six of the unary test queries almost satisfy

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

Then we can specify the fuzzy function $\tilde{\varphi}_{i2}$ in (6) at site i as follows

$$\begin{aligned} \tilde{\varphi}_{i2}(\widetilde{|R|}, \tilde{S}_\sigma) &= \{0.2/2.3,\ 0.5/5.8,\ 0.3/4.1\} \\ &\tilde{+}\ \{0.2/0.02,\ 0.5/0.1,\ 0.3/0.07\} \tilde{*} \widetilde{|R|} \\ &\tilde{+}\ \{0.2/0.003,\ 0.5/0.09,\ 0.3/0.02\} \tilde{*} \tilde{S}_\sigma \tilde{*} \widetilde{|R|} . \end{aligned}$$

Although there is a difference between possibility distribution and probability distribution, they have some relationship (see the possibility/probability consistency principle in Zadeh[26]). The above statistical method combined with the other two methods described below can give a reasonable fuzzy cost function.

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

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

that is,

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

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

$$\widetilde{|R|} = \{\frac{n_1}{n}/r_1,\ \frac{n_2}{n}/r_2,\ \ldots,\ \frac{n_t}{n}/r_t\} .$$

Other types of experiments are also possible.

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

Although some internal information about an object, for example, local database, local DBMS and network, may not be known, a fuzzy parameter can be constructed by using external characteristics of the object, such as capability and processing speed, which may be obtained from 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 a fuzzy cost function φ_{ik} in (6) at site i is known, then the experts may estimate the cost for processing a unary query at site j to be approximately 10 times of that at site i. One possible fuzzy cost function φ_{jk} at site j is

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

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 $\widetilde{Dl}_{jk}$ and $\widetilde{Dl}_{ik}$ $(l = 0, 1, 2)$, which reflect the experts' beliefs about the relationship between the costs at the two sites. The experts have probably taken into consideration other factors, such as site loads, in determining the fuzzy proportional constants. An expert may combine this method with the previous method in specifying a fuzzy cost function. In other words, An expert may adjust a fuzzy cost function obtained from experiments according to his/her previous knowledge.

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

Experts may make errors in constructing fuzzy parameters for a 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 cost 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, $\widetilde{E0}_{ik} = 3.56$ (i.e., $\{1.0/3.56\}$). 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 local DBMSs in the MDBS is fuzzy from 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 all fuzzy parameters are finite; that is, the experts give only a finite number of guesses for each fuzzy set.

4. Fuzzy Query Optimization

Establishing a fuzzy cost model for an MDBS is not our ultimate purpose. Our 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 (global) query optimization based on a fuzzy cost model.

4.1. *Optimization problem*

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 (estimate) $\tilde{c}_i$ can be calculated by using the fuzzy cost model in the last section. 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, to find x^0 such that

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

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

To define the $''min''$ operator, we need to define the ordering between two fuzzy costs. There are several ways to define the ordering. Different definitions imply different decisions about how to make use of fuzzy costs to find a possibly optimal execution strategy for a query. In other words, they represent different optimization criteria.

A simple way to define the ordering between two fuzzy costs $\tilde{c}_1$ and $\tilde{c}_2$ is as follows. $\tilde{c}_1 \leq \tilde{c}_2$ if and only if the average value of the elements with the highest grade of membership in $\tilde{c}_1$ is less or equal to the average value of the elements with the highest grade of membership in $\tilde{c}_2$. Although this ordering is intuitively correct because a real cost has the highest possibility to be one of the elements with the highest grade of membership in the fuzzy cost, it 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 adopt the following definition for the ordering between two fuzzy costs:

Definition 4 *Let $\tilde{c}_1$ and $\tilde{c}_2$ be two fuzzy costs. $\tilde{c}_1 \leq \tilde{c}_2$ if and only if $\omega(\tilde{c}_1) \leq \omega(\tilde{c}_2)$, where $\omega(\ )$ is the weighted average value of a fuzzy cost defined in (3).*

In this definition, all elements affect the ordering but in different weights according to their possibilities.

Using this definition of ordering, we can define $\underset{x \in X}{''min''}\ \ \tilde{F}(x)$ as a fuzzy cost (may not be unique) such that

$$\omega(\underset{x \in X}{''min''}\quad \tilde{F}(x))\ \ \leq\ \ \omega(\tilde{F}(y))\ ,\quad \forall y \in X\ . \tag{9}$$

The fuzzy query optimization criterion given in (8) and (9) can be used to find a possibly optimal strategy for a given query.

Example 1 For a given query, let $x_1,\ x_2,\ x_3$ and x_4 be all feasible execution strategies. Let $\tilde{c}_1\ =\ \{0.4/46.3,\ 0.8/65.8,\ 0.7/120.8\}$, $\tilde{c}_2\ =\ \{0.7/78.1,\ 0.9/75.2,$

0.6/34.8, 0.3/21.0}, $\tilde{c}_3 = \{0.5/52.9,\ 0.7/62.3,\ 0.7/127.3,\ 0.7/219.3,\ 0.4/394.1\}$ and $\tilde{c}_4 = \{0.9/542.0,\ 0.7/359.9\}$ be the fuzzy costs for x_1, x_2, x_3 and x_4, respectively. According to the definition of $''min''$ in (9), the strategy x_2 is the best, because

$$\begin{aligned}
\omega(\tilde{c}_2) = \frac{1}{2.5}&[0.7*78.1+0.9*75.2+0.6*34.8+0.3*21.0] = 59.8 \\
< \quad \omega(\tilde{c}_1) &= \frac{1}{1.9}[0.4*46.3+0.8*65.8+0.7*120.8] = 82.0 \\
< \quad \omega(\tilde{c}_3) &= \frac{1}{3.0}[0.5*52.9+0.7*62.3+0.7*127.3+0.7*219.3+0.4*394.1] \\
&= 156.8 \\
< \quad \omega(\tilde{c}_4) &= \frac{1}{1.6}[0.9*542.0+0.7*359.9] = 462.3\ .
\end{aligned}$$

In fact, the above fuzzy optimization criterion induces a fuzzy set that represents the fuzzy notion "good execution strategy", that is,

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

The membership space of $\tilde{G}$ is a subset of $(-\infty,\ 0]$ instead of $[0,\ 1]$. For each feasible strategy, $\tilde{G}$ assigns an assessment of goodness to it. In general, the higher the degree of goodness, the better the corresponding strategy. 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.

4.2. *Relationship between fuzzy and crisp approaches*

Let us now consider the relationship between the fuzzy optimization approach and the conventional crisp optimization approach. From the relationship, we can see why the fuzzy approach has a higher chance to be better than the 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 use a value α_1 that is of the highest possibility to be true from our guess as an estimate for C. If we are allowed to use a fuzzy set to describe C, we can specify not only α_1 but also other values with lower possibilities, that is, $C = \{\chi(\alpha_1)/\alpha_1,\ \chi(\alpha_2)/\alpha_2,\ \ldots,\ \chi(\alpha_n)/\alpha_n\}$ where $\chi(\alpha_1) \geq \chi(\alpha_i)$ $(1 < i \leq n)$.

For a fuzzy value (set) $\tilde{A} \tilde{\subset} X$, let

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

that is, $hgv(\tilde{A})$ is the set of elements with the highest grade in $\tilde{A}$. We have the following lemma:

Lemma 1 *Let $\tilde{A}_1$ and $\tilde{A}_2$ be two fuzzy values, then*

$$\forall x \ \forall y \ (x \in hgv(\tilde{A}_1) \ \wedge \ y \in hgv(\tilde{A}_2) \quad \Rightarrow \quad x \ \odot \ y \in hgv(\tilde{A}_1 \ \tilde{\odot} \ \tilde{A}_2) \ \ ) \ ,$$

where $\odot \in \{+,\ -,\ *\}$.

Proof. Let $\tilde{A}_1 \tilde{\subset} X_1$ and $\tilde{A}_2 \tilde{\subset} X_2$. For any $x_0 \in hgv(\tilde{A}_1)$ and $y_0 \in hgv(\tilde{A}_2)$, we have a defined value $x_0 \odot y_0$. Since $\chi_{\tilde{A}_1}(x_0) = \sup_{x \in X_1}\{\chi_{\tilde{A}_1}(x)\}$ and $\chi_{\tilde{A}_2}(y_0) = \sup_{x \in X_2}\{\chi_{\tilde{A}_2}(x)\}$, we have

$$\begin{aligned} min\{\chi_{\tilde{A}_1}(x_0),\ \chi_{\tilde{A}_2}(y_0)\} &= min\{\sup_{x \in X_1}\{\chi_{\tilde{A}_1}(x)\},\ \sup_{x \in X_2}\{\chi_{\tilde{A}_2}(x)\}\} \\ &\geq min\{\chi_{\tilde{A}_1}(x_1),\ \chi_{\tilde{A}_2}(y_1)\}\ , \end{aligned}$$

for any $x_1 \in X_1$ and $y_1 \in X_2$. Therefore,

$$\begin{aligned} &\chi_{\tilde{A}_1 \tilde{\odot} \tilde{A}_2}(x_0 \odot y_0) \ = \ [\sup_{(x,y) \in X_1 \times X_2,\ x \odot y = x_0 \odot y_0} min\{\chi_{\tilde{A}_1}(x), \chi_{\tilde{A}_2}(y)\}\ ] \\ &= \ min\{\chi_{\tilde{A}_1}(x_0),\ \chi_{\tilde{A}_2}(y_0)\} \ \geq \ [\sup_{(x,y) \in X_1 \times X_2,\ x \odot y = z} min\{\chi_{\tilde{A}_1}(x), \chi_{\tilde{A}_2}(y)\}\ ] \\ &= \ \chi_{\tilde{A}_1 \tilde{\odot} \tilde{A}_2}(z)\ , \end{aligned}$$

for any $z \in \{z \mid z = x \odot y,\ (x, y) \in X_1 \times X_2\}$. In other words, $x_0 \odot y_0 \in hgv(\tilde{A}_1\ \tilde{\odot}\ \tilde{A}_2)$. $\square$

Using Lemma 1, we can show the following theorem.

Theorem 1 *Let c be a crisp cost of an execution strategy x for a query. It is derived from the fuzzy cost model in Section 3 by using a crisp estimate for each parameter. If every parameter in the fuzzy cost model is replaced by a fuzzy estimate that includes the corresponding crisp estimate used to derive c as one of its elements with the highest grade of membership, and let $\tilde{C}$ be a fuzzy cost of x derived by using the fuzzy estimates, then c is one of the elements with the highest grade of membership in $\tilde{C}$, that is, $c \in hgv(\tilde{C})$.*

Proof. Notice that the fuzzy operations used in the fuzzy cost model are $\tilde{+}$, $\tilde{-}$, and $\tilde{*}$. By structural induction using Lemma 1 for the cost model in Section 3, it is not difficult to see that $c \in hgv(\tilde{C})$. $\square$

Theorem 1 states that a crisp cost derived by using crisp estimates is always contained in a fuzzy cost derived by using fuzzy estimates. A fuzzy cost, therefore, contains more information than a crisp cost. The other information contained in the fuzzy cost is lost from the crisp cost. Because the fuzzy optimization approach has more information to make use of, it has a higher chance to be successful in choosing a good strategy than a crisp approach.

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

s_1**:** transfer R_1 from site 1 to site 2 and perform the join at site 2.

s_2**:** transfer R_2 from site 2 to site 1 and perform the join at site 1.

For simplicity, assume that the local processing costs at both sites can be neglected, compared with the communication costs; that is, $\widetilde{Ej}_{ik} = 0$ $(0 \leq j \leq 4; i = 1, 2)$ in a cost function in (7) for an employed join access method k. Hence the costs for executing s_1 and s_2 are determined by the communication costs that are computed by using the cost function in (5). We want to choose a cheaper strategy between s_1 and s_2. The real values, fuzzy estimates, and crisp estimates for the coefficients and inputs in (5) are given in Table 1. The real values may not be known by the

Table 1. Parameter Values for a Fuzzy Cost Function

parameter	real value	fuzzy estimate	crisp estimate
$\widetilde{C0}_{12}$	3.5	{0.5/2.8, 0.8/3.5, 0.3/5.5}	3.5
$\widetilde{C1}_{12}$	0.0004	{0.5/0.0001, 0.7/0.0002, 0.9/0.0008}	0.0008
$\widetilde{\lvert R_1 \rvert}$	850	{0.5/575, 0.7/1200}	1200
tl_1	546	546	546
$\widetilde{\lvert R_2 \rvert}$	1400	{0.3/590, 0.9/900, 0.7/1790}	900
tl_2	480	480	480
$\tilde{x}_1 = \widetilde{\lvert R_1 \rvert} \tilde{*} tl_1$	464100	{0.5/313950, 0.7/655200}	655200
$\tilde{x}_2 = \widetilde{\lvert R_2 \rvert} \tilde{*} tl_2$	672000	{0.3/283200, 0.9/432000, 0.7/859200}	432000
tl_1 — tuple length of R_1, tl_2 — tuple length of R_2			

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

Using real values in Table 1, we have

$$\tilde{\delta}_{12}(\tilde{x}_1) = 3.5 + 0.0004 * 464100 = 189.1 ,$$

and

$$\tilde{\delta}_{21}(\tilde{x}_2) = 3.5 + 0.0004 * 672000 = 272.3 ,$$

which are the real costs for executing s_1 and s_2, respectively. Since $\tilde{\delta}_{12}(\tilde{x}_1) < \tilde{\delta}_{21}(\tilde{x}_2)$, strategy s_1 is more efficient than strategy s_2.

Using fuzzy estimates in Table 1, we have

$$\begin{aligned}\tilde{\delta}_{12}(\tilde{x}_1) = \{&0.5/2.8,\ 0.8/3.5,\ 0.3/5.5\} \tilde{+} \\ &\{0.5/0.0001,\ 0.7/0.0002,\ 0.9/0.0008\} \tilde{*} \{0.5/313950,\ 0.7/655200\} \\ = \{&0.5/34.2,\ 0.5/34.9,\ 0.3/36.9,\ 0.5/65.6,\ 0.5/66.3,\ 0.5/68.3, \\ &0.5/69.0,\ 0.3/71.0,\ 0.5/133.8,\ 0.7/134.5,\ 0.3/136.5,\ 0.5/254.0, \\ &0.5/254.7,\ 0.3/256.7,\ 0.5/527.0,\ 0.7/527.7,\ 0.3/529.7\} ,\end{aligned}$$

and

$$\begin{aligned}\tilde{\delta}_{21}(\tilde{x}_2) = \{&0.5/2.8,\ 0.8/3.5,\ 0.3/5.5\} \tilde{+} \\ &\{0.5/0.0001,\ 0.7/0.0002,\ 0.9/0.0008\} \tilde{*} \{0.3/283200,\ 0.9/432000,\ 0.7/859200\} \\ = \{&0.3/31.1,\ 0.3/31.8,\ 0.3/33.8,\ 0.5/46.0,\ 0.5/46.7,\ 0.3/48.7,\ 0.3/59.4, \\ &0.3/60.1,\ 0.3/62.1,\ 0.5/88.7,\ 0.5/89.2,\ 0.5/89.4,\ 0.7/89.9,\ 0.3/91.4, \\ &0.3/91.9,\ 0.5/174.6,\ 0.7/175.3,\ 0.3/177.3,\ 0.3/229.4,\ 0.3/230.1,\ 0.3/232.1, \\ &0.5/348.4,\ 0.8/349.1,\ 0.3/351.1,\ 0.5/690.2,\ 0.7/690.9,\ 0.3/692.9\} .\end{aligned}$$

By the definition of $''min''$ in (9), we choose s_1 as a better strategy because $\omega(\tilde{\delta}_{12}(\tilde{x}_1)) = 193.3 \ < \ \omega(\tilde{\delta}_{21}(\tilde{x}_2)) = 216.0$, which is correct.

Using the crisp estimates in Table 1, we have

$$\begin{aligned} \tilde{\delta}_{12}(\tilde{x}_1) \ &= \ 3.5 \ + \ 0.0008 \ * \ 655200 \ = \ 527.7 \\ > \ \tilde{\delta}_{21}(\tilde{x}_2) \ &= \ 3.5 \ + \ 0.0008 \ * \ 432000 \ = \ 349.1 \ , \end{aligned}$$

which gives a wrong conclusion that s_2 is better. The reason for this error is that the crisp parameter estimates used are not accurate. A fuzzy parameter estimate reflects experts' spontaneous fuzzy perception of an MDBS. Errors made in a forced crisp parameter estimate can be adjusted by the values with lower grades of membership in the fuzzy parameter estimate. Hence a fuzzy approach has a higher chance to be successful in choosing a good execution strategy for a query than a crisp one when a correct crisp choice is difficult to obtain, such as in an MDBS environment. □

Although the fuzzy query optimization approach has a better chance to find a good execution strategy for a query in an MDBS, a problem with it is that its computing complexity may be much higher than the crisp query optimization approach. We have the following theorem:

Theorem 2 *Given an execution strategy x for a query, if using crisp estimates for all parameters in the fuzzy cost model in Section 3 to calculate a crisp cost for x requires n arithmetic operations, then using fuzzy estimates for all parameters in the fuzzy cost model to calculate a fuzzy cost $\tilde{C}$ for x requires $O(b^n)$ (crisp) arithmetic operations where*

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

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

Proof. Notice that if n crisp arithmetic operations are used in the cost model in the crisp case, then n fuzzy (arithmetic) operations are used in the cost model in the fuzzy case.

Let p be the number of (crisp) arithmetic operations required to perform n fuzzy operations and q be the cardinality of the support of the final fuzzy result. We first prove that

$$p \leq \sum_{i=2}^{n+1} b^i \tag{10}$$

and

$$q \leq b^{n+1} \tag{11}$$

by using mathematical induction on n.

(i) For $n = 1$, there is one fuzzy operation on two fuzzy operands. By the Extension Principle, we know that at most b^2 (crisp) arithmetic operations are required to perform the fuzzy operation and there are at most b^2 elements with non-zero grades in the fuzzy result; that is, inequalities (10) and (11) hold.

(ii) Assume that inequalities (10) and (11) are true for any $n \leq k-1$. For $n = k$, there are two cases:

(a) One of the operands for the last fuzzy operation is the result of $k-1$ fuzzy operations, the other operand is a fuzzy parameter estimate. From the inductive assumption of (11) for $n \leq k-1$ and the Extension Principle, at most $b^k * b = b^{k+1}$ arithmetic operations required to perform the last fuzzy operation, and there are at most b^{k+1} elements with non-zero grades in the final fuzzy result. From the inductive assumption of (10) for $n \leq k-1$,

$$p \leq \sum_{i=2}^{k} b^i + b^{k+1} = \sum_{i=2}^{k+1} b^i \ .$$

(b) Two operands of the last fuzzy operation are the results of m fuzzy operations and $k-1-m$ fuzzy operations, respectively, where $1 \leq m < k-1$. From the inductive assumption of (11) for $n \leq k-1$ and the Extension Principle, at most $b^{m+1} * b^{k-m} = b^{k+1}$ arithmetic operations required to perform the last fuzzy operation, and there are at most b^{k+1} elements with non-zero grades in the final fuzzy result. From the inductive assumption of (10) for $n \leq k-1$,

$$p \leq \sum_{i=2}^{m+1} b^i + \sum_{j=2}^{k-m} b^j + b^{k+1} \ . \tag{12}$$

Since $b > 1$ and $m \geq 1$,

$$\begin{aligned}\sum_{j=2}^{k-m} b^j &= \frac{b^{k-m+1} - b^2}{b-1} = \frac{1}{b^m} * \frac{b^{k+1} - b^{m+2}}{b-1} \\ &\leq \frac{b^{k+1} - b^{m+2}}{b-1} = \sum_{i=m+2}^{k} b^i \ . \end{aligned}\tag{13}$$

From (12) and (13), we have

$$p \leq \sum_{i=2}^{m+1} b^i + \sum_{i=m+2}^{k} b^i + b^{k+1} = \sum_{i=2}^{k+1} b^i \ . \tag{14}$$

From (i) and (ii), we conclude that (10) and (11) are true for all n. From (10), we have

$$p \ \leq \ \sum_{i=2}^{n+1} b^i = \frac{b^2 * (b^n - 1)}{b-1} \ .$$

Therefore, the claim in the theorem is true. □

In practice, the computing complexity of the fuzzy query optimization approach may be lower if most of the fuzzy parameter estimates are actually crisp or many fuzzy operations yield fewer elements in their results because of the removal of duplicate elements during computation. However, in general, the computing complexity of the fuzzy approach can be as high as b^n. In other words, the complexity may grow exponentially as n increases.

5. Reducing Computing Complexity

In order to reduce the complexity, the result size of a fuzzy operation should be controlled. As we will see below, if we restrict the result size of each fuzzy operation within a fixed range, the complexity of computing n fuzzy operations will not grow exponentially.

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. 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 lower grades of membership, that is, setting their grades of membership to zeros.

Definition 5 *Let* $\tilde{A} = \{d_1/a_1,\ d_2/a_2,\ ...,\ d_{s-1}/a_{s-1},\ d_s/a_s,\ ...,\ d_k/a_k,\ d_{k+1}/a_{k+1},\ ...,\ d_t/a_t,\ d_{t+1}/a_{t+1},\ ...,\ d_m/a_m\}$ *be a fuzzy value, where* $d_1 \geq d_2 \geq ... \geq d_{s-1} > d_s = ... = d_k = d_{k+1} = ... = d_t > d_{t+1} \geq ... \geq d_m$; $a_s \geq ... \geq a_k \geq a_{k+1} \geq ... \geq a_t$; $1 \leq s \leq k$, *and* $k \leq t \leq m$. *A* k*-approximate fuzzy value for* $\tilde{A}$ *is defined as* $\tilde{A}^{(k)} = \{d_1/a_1,\ d_2/a_2,\ ...,\ d_k/\bar{a}_k\}$, *where* $\bar{a}_k = (\sum_{i=k}^{t} a_i)/(t-k+1)$. *If* $k > m$, *we define* $\tilde{A}^{(k)} = \tilde{A}$.

For example, $\tilde{A}^{(2)} = \{0.9/6,\ 0.7/4\}$, $\tilde{A}^{(3)} = \{0.9/6,\ 0.7/4,\ 0.5/5\}$, and $\tilde{A}^{(7)} = \{0.9/6,\ 0.7/4,\ 0.5/7,\ 0.5/3,\ 0.3/9\}$ are the 2-approximate, 3-approximate, and 7-approximate fuzzy values for the fuzzy value $\tilde{A} = \{0.9/6,\ 0.7/4,\ 0.5/7,\ 0.5/3,\ 0.3/9\}$.

The reason for sorting $a_s,\ ..., a_k,\ a_{k+1},\ ...,\ a_t$ in the decreasing order is to give higher priorities of consideration to larger values of the same grade of membership so that the larger cost estimates of the same possibility are favored.[5] When not all values of the same grade of membership can be taken into an approximate fuzzy value, an average value is used to aggregate smaller values of the same grade of membership that would be completely lost otherwise.

If we use a k-approximate fuzzy value to approximate every fuzzy parameter and the result of every fuzzy operation in the fuzzy cost model, the following theorem shows that the computing complexity of the fuzzy query optimization approach is

[5]Note that the practical goal of many query optimizers is to avoid bad execution strategies for a query, i.e., it needs to identify the execution strategies with large costs.

of the same order as that of the crisp one.

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

$$b = max\{\ |Supp(\tilde{c})|\ \ |\ \tilde{c}\ is\ a\ fuzzy\ parameter\ estimate\ used\ in\ the\ computation\ \}\ .$$

Assume $b \geq k \geq 1$.

Proof. Notice that if n crisp (binary) arithmetic operations are used in the fuzzy cost model in the crisp case, n fuzzy (binary) arithmetic operations are used in the fuzzy case.

A computation involving binary operations can be represented by a full binary tree whose internal nodes represent binary operations and leaves represent the original inputs for the computation. From a theorem in graph theory,[27] a full binary tree with n internal nodes has $n+1$ leaves. Hence there are $n+1$ fuzzy parameter estimates as inputs for the computation involving n fuzzy arithmetic operations. By Definition 5, at most $b-k+3$ crisp arithmetic operations are needed to calculate a k-approximate fuzzy value for a fuzzy parameter estimate. Therefore, at most $(n+1)*(b-k+3)$ crisp arithmetic operations are needed to calculate the k-approximate fuzzy values for the $(n+1)$ fuzzy parameter estimates required for the n fuzzy arithmetic operations.

Since we use k-approximate fuzzy values to approximate all fuzzy parameter estimates and the results of all fuzzy operations during the fuzzy computation, every fuzzy operation operates on two k-approximate fuzzy values. By the Extension Principle, the number of crisp arithmetic operations required to perform such a fuzzy operation is k^2. The number of crisp arithmetic operations required to approximate the result of the fuzzy operation by a k-approximate fuzzy value is at most k^2-k+3. Thus, the total number of crisp arithmetic operations required to perform the n fuzzy operations and approximate the n results by k-approximate fuzzy values is at most $n*(2*k^2-k+3)$.

Therefore, at most $(n+1)*(b-k+4)+n*(2*k^2-k+3) = O(n*(b+2*k^2))$ crisp arithmetic operations are required for the computation involving n fuzzy operations. □

Since b is a finite number in our cost model and $b \geq k$, the complexity of the fuzzy query optimization approach is of the same order of the complexity as the corresponding crisp one. If $k=1$, the fuzzy query optimization approach is reduced to the crisp one. By choosing an appropriate $k > 1$, we can get an efficient fuzzy optimization method. One of the reasonable choices is setting $k = b$.

Example 3 In Example 2, taking 3-approximate fuzzy values for all fuzzy parameter estimates and fuzzy results, we have

$$\begin{aligned}\tilde{\delta}_{12}(\tilde{x}_1)^{(3)} &= [\{0.5/2.8,\ 0.8/3.5,\ 0.3/5.5\}\ \tilde{+} \\ &\qquad (\{0.5/0.0001,\ 0.7/0.0002,\ 0.9/0.0008\}\ \tilde{*}\ \{0.5/313950,\ 0.7/655200\})^{(3)}]^{(3)} \\ &= [\{0.5/2.8,\ 0.8/3.5,\ 0.3/5.5\}\ \tilde{+}\ \{0.7/524.2,\ 0.7/131.0,\ 0.5/102.7\}]^{(3)} \\ &= \{0.7/527.0,\ 0.7/134.5,\ 0.5/218.1\}\ ,\end{aligned}$$

and

$$\begin{aligned}\tilde{\delta}_{21}(\tilde{x}_2)^{(3)} &= [\{0.5/2.8,\ 0.8/3.5,\ 0.3/5.5\}\ \tilde{+}\ (\{0.5/0.0001,\ 0.7/0.0002, \\ &\qquad 0.9/0.0008\}\ \tilde{*}\ \{0.3/283200,\ 0.9/432000,\ 0.7/859200\})^{(3)}]^{(3)} \\ &= [\{0.5/2.8,\ 0.8/3.5,\ 0.3/5.5\}\ \tilde{+}\ \{0.9/345.6,\ 0.7/687.4,\ 0.7/129.1\}]^{(3)} \\ &= \{0.8/349.1,\ 0.7/690.9,\ 0.7/132.6\}\ .\end{aligned}$$

Since $\omega(\tilde{\delta}_{12}(\tilde{x}_1)^{(3)}) = 301.4 < \omega(\tilde{\delta}_{21}(\tilde{x}_2)^{(3)}) = 389.0$, s_1 is chosen as a better strategy. This choice agrees with the one made in Example 2. □

Clearly, using 3-approximate fuzzy values can reduce a large amount of computation and still keep most information from the original fuzzy values. It, therefore, has the advantages of both fuzzy and crisp query optimization approaches.

6. Conclusion

How to perform global query optimization when some required optimization information is not accurately known at the global level in an MDBS is a new challenge. This paper presents a novel 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 perception 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 where precise information is not available, the fuzzy query optimization approach has a better chance to be successful 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 in this paper 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 by experiments, external characteristics of objects, and runtime information. It is shown that a fuzzy cost estimate using a fuzzy cost model contains more information than a crisp cost estimate using a crisp cost model. A proper criterion is suggested for query optimization using a fuzzy cost model.

However, it is shown that the computing complexity of a raw fuzzy query optimization approach may grow exponentially as n, the computing complexity of the corresponding crisp query optimization approach, grows. To reduce the complexity of the fuzzy approach, a k-approximate fuzzy value is suggested to approximate each fuzzy value used in fuzzy optimization. The fuzzy optimization approach using such k-approximate fuzzy values is proven to have the same order of complexity as the corresponding crisp approach. Therefore fuzzy optimization is shown to be computationally feasible.

It should be pointed out that a crisp cost model is a special case of a fuzzy cost model. When precise information is available, the fuzzy cost model boils down to a crisp one. There is no difference between the fuzzy query optimization approach and the crisp one in this case. When only fuzzy information is available, the fuzzy approach can make better use of such information, while forced exactness in the crisp approach may mislead the global query optimizer into choosing a bad execution strategy for a query. However, when the parameters in a cost model can be estimated quite accurately, the crisp query optimization approach, i.e., using 1-approximate fuzzy values, is recommended due to its high efficiency.

We plan to further explore other methods to establish a good fuzzy cost model for an MDBS and develop more algorithms for efficient fuzzy query optimization, e.g., fuzzy dynamic programming. We also plan to develop some heuristics based on fuzzy information to study the feasibility of heuristic-based fuzzy query optimization.

Acknowledgments

The authors would like to thank the anonymous referees for their careful reading and valuable comments. This research was supported by the IBM Canada Ltd. Laboratory, the Natural Sciences and Engineering Research Council (NSERC) of Canada, and the University of Waterloo.

References

1. M. W. Bright, A. R. Hurson, and S. H. Pakzad, "A taxonomy and current issues in multidatabase systems", *IEEE Computer* **25** (1976) 50–59.
2. H. Lu, B.-C. Ooi, and C.-H. Goh, "On global multidatabase query optimization", *SIGMOD Record* **21** (1992) 6–11.
3. H. Lu and M.-C. Shan, "On global query optimization in multidatabase systems", *Proc. 2nd Int. workshop on Research Issues on Data Eng.*, Tempe, Arizona, USA, 1992, p. 217.
4. Qiang Zhu, "Query optimization in multidatabase systems", *Proc. 1992 IBM CAS Conf.*, Toronto, Canada, Nov. 1992, vol.II, pp. 111–127
5. Qiang Zhu, "An integrated method of estimating selectivities in a multidatabase system", *Proc. 1993 IBM CAS Conf.*, Toronto, Canada, Oct. 1993, pp. 832–847.
6. Qiang Zhu and P.-Å. Larson, "Global query processing and optimization in the CORDS multidatabase system", *Proc. 9th ISCA Int. Conf. on Parallel and Distributed Computing Systems*, Dijon, France, Sept. 1996, pp. 640–646.
7. W. Du, R. Krishnamurthy, and M. C. Shan, "Query optimization in heterogeneous DBMS", *Proc. 18th Int. Conf. on Very Large Data Bases*, Vancouver, Canada, Aug. 1992, pp. 277–291.

8. Qiang Zhu and P.-Å. Larson, "Building regression cost models for multidatabase systems", *Proc. 4th IEEE Int. Conf. on Parallel and Distributed Information Systems*, Miami Beach, Florida, Dec. 1996, pp. 220–231.
9. Qiang Zhu and P.-Å. Larson, "A query sampling method for estimating local cost parameters in a multidatabase system", *Proc. 10th IEEE Int. Conf. on Data Engineering*, Houston, Texas, Feb. 1994, pp. 144–153.
10. G.K. Attaluri, D.P. Bradshaw, N. Coburn, P.-Å. Larson, P. Martin, A. Silberschatz, J. Slonim, and Qiang Zhu, "The CORDS multidatabase project", *IBM Systems Journal* **34** (1995) 39–62.
11. L. A. Zadel, "Fuzzy sets", *Inform. and Control* **8** (1965) 338–353.
12. F. E. Petry, *Fuzzy Databases: Principles and Applications* (Kluwer Academic Publishers, 1996).
13. B. Buckles and F. Petry, "A fuzzy model for relational databases", *Fuzzy Sets and Systems* **7** (1982) 213–226.
14. B. Buckles, F. Petry, and J. Pillai, "Network data models for representation of uncertainty", *Fuzzy Sets and Systems* **38** (1990) 171–190.
15. T. Imieminski and W. Lipski, "Incomplete information in relational databases", *J. of the ACM* **31** (1984) 761–791.
16. M. Zemankova and A. Kandel, "Implementing imprecision in information systems", *Information Sciences* **37** (1985) 107–141.
17. R. Zicari, "Incomplete information in object-oriented databases", *SIGMOD RECORD* **19** (1990) 33–40.
18. P. Bosc and O. Pivert, "SQLf: A relational database language for fuzzy querying", *IEEE Transaction on Fuzzy Systems* **3** (1995) 1–17.
19. B. Buckles and F. Petry, "Query languages for fuzzy databases", in *Management Decision Support Systems Using Fuzzy Sets and Possibility Theory*, eds. J. Kacprzyk and R. Yager (1985) pp. 241–252.
20. S. Pak et al., "Fuzzy querying in relational databases", *Proc. 5th IFSA World Congress*, 1993, pp. 553–536.
21. Y. Takahashi, "A fuzzy query language for relational databases", *IEEE Trans. on Systems, Man and Cybernetics* **21** (1991) 1576–1579.
22. P. Saxena and B. Tyagi, "Fuzzy functional dependencies and independencies in extended fuzzy relational database models", *Fuzzy Sets and Systems* **69** (1995) 65–89.
23. D. Lee and M. Kim, "Accommodating subjective vagueness through a fuzzy extension to the relational data model", *Information Systems* **18** (1993) 363–374.
24. P. Bosc and M. Galibourg, "Indexing principles for a fuzzy data base", *Information Systems* **14** (1989) 493–499.
25. V. Novak, *Fuzzy Sets and Their Applications* (Adam Hilger, 1989).
26. L.A. Zadeh, "Fuzzy sets as a basis for a theory of possibility", *Fuzzy Sets and Systems* **1** (1978) 3–28.
27. R. Johnsonbaugh, *Discrete Mathematics* (Macmillan Publishing Company, 1993).