Developing Cost Models with Qualitative Variables for Dynamic Multidatabase Environments*

Qiang Zhu Yu Sun S. Motheramgari
Department of Computer and Information Science
The University of Michigan, Dearborn, MI 48128, U.S.A.
{qzhu, yusun, motheram}@umich.edu

Abstract

A major challenge for global query optimization in a multidatabase system (MDBS) is lack of local cost information at the global level due to local autonomy. A number of methods to derive local cost models have been suggested recently. However, these methods are only suitable for a static multidatabase environment. In this paper, we propose a new multi-states query sampling method to develop local cost models for a dynamic environment. The system contention level at a dynamic local site is divided into a number of discrete contention states based on the costs of a probing query. To determine an appropriate set of contention states for a dynamic environment, two algorithms based on iterative uniform partition and data clustering, respectively, are introduced. A qualitative variable is used to indicate the contention states for the dynamic environment. The techniques from our previous (static) query sampling method, including query sampling, automatic variable selection, regression analysis, and model validation, are extended so as to develop a cost model incorporating the qualitative variable for a dynamic environment. Experimental results demonstrate that this new multi-states query sampling method is quite promising in developing useful cost models for a dynamic multidatabase environment.

1. Introduction

A multidatabase system (MDBS) integrates data from multiple local (component) databases and provides users with a uniform global view of data. A global user can issue a (global) query on an MDBS to retrieve data from multiple databases without having to know where the data is stored and how the data is retrieved. How to process such a global query efficiently is the task of global query optimization.

*Research supported by the US National Science Foundation under Grant # IIS-9811980 and The University of Michigan under OVPR and UMD grants.

A major challenge, among others [4, 7, 8, 9, 14], for global query optimization in an MDBS is that some necessary local information, such as local cost models, may not be available at the global level due to local autonomy preserved in the system. However, the global query optimizer needs such information to decide how to decompose a global query into local (component) queries and where to execute the local queries. Hence, methods to derive cost models for an autonomous local database system (DBS) at the global level are required. Several such methods have been proposed in the literature recently.

In [3], Du *et al.* proposed a calibration method to deduce necessary local cost parameters. The key idea is to construct a special local synthetic calibrating database and use the costs of some special queries run on the database to deduce the parameters in cost models. In [5], Gardarin *et al.* extended the above method so as to calibrate cost models for object-oriented local database systems in an MDBS.

In [17, 18, 19], Zhu and Larson proposed a query sampling method. The key idea is as follows. It first groups local queries that can be performed on a local DBS in an MDBS into homogeneous classes, based on some information available at the global level in an MDBS such as the characteristics of queries, operand tables and the underlying local DBS. A sample of queries are then drawn from each query class and run against the user local database. The costs of sample queries are used to derive a cost model for each query class by multiple regression analysis. The cost model parameters are kept in the MDBS catalog and utilized during query optimization. To estimate the cost of a local query, the class to which the query belongs is first identified. The corresponding cost model is retrieved from the catalog and used to estimate the cost of the query. Based on the estimated local costs, the global query optimizer chooses a good execution plan for a global query.

There are several other approaches to tackling this problem. In [16], Zhu and Larson introduced a fuzzy method based on fuzzy set theory to derive cost models in an MDBS. In [10], Naacke *et al.* suggested an approach to combining a generic cost model with specific cost information exported by wrappers for local DBSs. In [1], Adali *et al.* suggested to maintain a cost vector database to record cost information for every query issued to a local DBS. Cost estimation for a new query is based on the costs of similar queries. In [13], Roth *et al.* introduced a framework for costing in the *Garlic* federated system.

All the methods proposed so far only considered a static system environment, i.e., assuming that it does not change significantly over time. However, in reality, many factors in an MDBS environment such as contention factors (e.g., number of concurrent processes), database physical characteristics (e.g., index clustering ratio), and hardware configurations (e.g., memory size) may change significantly over time. Hence, a cost model derived for a static system environment cannot give good cost estimates for queries in a dynamic environment. Figure 1 shows how the cost of a sample query is affected by the number of concurrent processes in a dynamic system environment. We can see that the cost of the same query can dramatically change (from 3.80 sec. to 124.02 sec.) in a dynamic environment. This raises an interesting research issue, that is, how to derive cost models that can capture the performance behavior of queries in a dynamic environment.

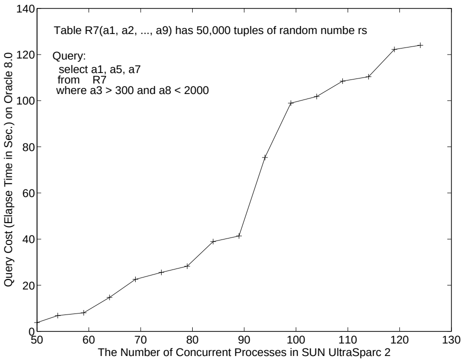

Figure 1. Effect of Dynamic Factor on Query Cost

In this paper, we propose a new qualitative approach to deriving a cost model that can capture the performance behavior of queries in a dynamic environment. We notice that there are numerous dynamic factors that affect query costs. To simplify the development of a cost model for a dynamic environment, our approach considers the combined net effect of dynamic factors on a query cost together rather than individually. The system contention level that reflects such a combined effect is gauged by the cost of a probing query. To capture such contention information in a cost model, we divide the system contention level (based on the costs of a probing query) in a dynamic environment into a number of discrete contention states and use a qualitative variable to indicate the contention states in the cost model. To determine an appropriate set of contention states for a dynamic environment, two algorithms, called the iterative uniform partition with merging adjustment (IUPMA) and the iterative clustering with merging adjustment (ICMA) respectively, are introduced. The former is used for general cases, while the latter is specifically designed for a dynamic environment with the contention level following a non-uniform distribution with clusters. Our previous query sampling method in [17, 18, 19] is extended so as to develop a regression cost model incorporating the qualitative variable for a dynamic environment. Our approach in this paper is therefore an extension of our previous query sampling method. In this paper, we call our previous method as the static query sampling method and the new approach in this paper as the multi-states query sampling method. In fact, the static method is a special case of the multi-states one when only one contention state is allowed.

The rest of the paper is organized as follows. Section 2 analyzes the dynamic factors at a local site in an MDBS. Section 3 discusses how to develop a regression model with a qualitative variable and how to determine contention states of a qualitative variable for a dynamic environment. Section 4 extends our previous static query sampling method so as to derive cost models with a qualitative variable for different query classes in a dynamic environment. Section 5 shows some experimental results. Section 6 summarizes the conclusions.

2. Dynamic environmental factors

In an MDBS, many environmental factors may change over time[1]. Some may change more often than others. They can be classified into the following three types based on their changing frequencies.

- *Frequently-changing factors*. The main characteristic of this type of factors is that they change quite often. Examples of such factors are CPU load, number of I/Os per second, and size of memory space being used, etc. The operating system at a local site typically provides commands (such as top, ps, and $iostats$ in Unix) to display system statistics reflecting such environmental factors. Table 1 lists some system statistics in Unix.

- *Occasionally-changing factors*. These factors change occasionally. Examples of such factors are local database management system (DBMS) configuration parameters (e.g., number of buffer blocks, and shared pool size), local database physical/conceptual schema (e.g., new indexes, new tables/columns), and local hardware configurations (e.g, physical memory size). Note that some other factors such as local database

[1] Since we concern ourselves with local cost models for an MDBS, only dynamic factors at local sites are considered. In general, there are also dynamic network environmental factors in an MDBS. Some of them were considered in [15]

Types	Statistics for Frequently-Changing Environmental Factors
CPU Statistics	• rp — number of running processes; • sp — number of sleeping processes • tp — number of stopped processes; • zp — number of zombie processes • us — percentage of user time; • sy — percentage of system time • id — percentage of idle time • ld_1, ld_2, ld_3 — load averages for the past 1, 5, and 15 minutes, respectively
Memory Statistics	• am — available memory; • um — used memory • sm — shared memory; • bm — buffer memory • as — available swap; • us — used swap • fs — free swap; • as — cached swap • si — amount of memory swapped in; • so — amount of memory swapped out
I/O Statistics	• bi — number of reads per sec.; • bo — number of writes per sec. • du — percentage of disk utilization
Other Statistics	• nu — number of current users; • in — number of interrupts per sec. • cs — number of context switches per sec.; • sc — number of system calls per sec.

Table 1. System Stats for Frequently-Changing Factors in Unix

size, physical data distribution, and index clustering ratio may change quite frequently. However, they may not have an immediate significant impact on query cost until such changes accumulate to a certain degree. Thus we also consider these factors as occasionally-changing factors. The changes of occasionally-changing factors can be found via checking the local database catalog and/or system configuration files.

- *Steady factors*. These factors rarely change. Examples of such factors are local DBMS type (e.g., relational or object-oriented), local database location (e.g., local or remote), and local CPU speed (e.g., 300MHz). Although these factors may have an impact on a cost model, the chance for them to change is very small.

Clearly, the steady factors usually do not cause a problem for a query cost model. If significant changes for such factors occur at a local site, they can be handled in a similar way as described below for the occasionally-changing factors.

For the occasionally-changing factors, a simple and effective approach to capturing them in a cost model is to invoke the static query sampling method periodically or whenever a significant change for the factors occurs. Since these factors do not change very often, rebuilding cost models from time to time to capture them is acceptable. However, this approach cannot be used for the frequently-changing factors because frequent invocations of the static query sampling method would significantly increase the system load and the cost model maintenance overhead. On the other hand, if a cost model cannot capture the dramatical changes in a system environment, poor query cost estimates may be used by the query optimizer, resulting in inefficient query execution plans.

Theoretically speaking, to capture the frequently-changing factors in a cost model, one approach is to include all explanatory variables that reflect such factors in the cost model. However, this approach encounters several difficulties. First, the ways in which these factors affect a query cost are not clear. As a result, the appropriate format of a cost model that directly includes the relevant variables is hard to determine. Second, the large number of such factors (see Table 1) makes a cost model too complicated to derive and maintain even if the previous difficulty could be overcome. In the rest of this paper, we introduce a feasible method to capture the frequently-changing factors in a cost model.

3. Regression with qualitative variable

As mentioned before, the key idea of our method is to determine a number of contention states for a dynamic environment and use a qualitative variable to indicate the states. A cost model with the qualitative variable can be used to estimate the cost of a query in different contention states. The issues on how to include a qualitative variable in a cost model and how to determine an appropriate set of system contention states are discussed in this section.

3.1. Qualitative variable

To simplify the problem, we consider the combined effect of all the frequently-changing factors on a query cost together rather than individually. Although these dynamic factors may change differently in terms of the changing frequency and degree, they all contribute to the contention level of the underlying system environment. The cost of a query increases as the contention level. The system contention level can be divided into a number of discrete states (categories) such as "$High\ Contention$" (S_H), "$Medium\ Contention$"(S_M), "$Low\ Contention$" (S_L), and "$No\ Contention$" (S_N). A qualitative variable W is used to indicate the contention states. This qualitative variable, therefore, reflects the combined effect of foregoing frequently-changing environmental factors. A cost model incorporating such a qualitative variable can capture the dynamic environmental factors to certain degree.

As shown in [17, 19], a statistical relationship between query costs and their affecting factors such as operand and result table sizes can be established by multiple regression. The established relationship can be then used as a cost model to estimate query costs.

Usually, only quantitative variables are considered in a regression model. These variables such as operand table size take values on a well-defined scale. However, many variables of interest may not be quantitative but qualitative. Qualitative variables only have several discrete categories (states). For example, the foregoing qualitative variable W indicating system contention states may have states S_H, S_M, S_L, and S_N. Such a qualitative variable can also be incorporated into a regression model.

A qualitative variable can be represented by a set of indicator variables. For example, the above contention state variable W with four states can be represented by three indicator variables: Z_1, Z_2, and Z_3, where $Z_1 = 1$ indicates $W = S_H$, while $Z_1 = 0$ indicates $W \neq S_H$; $Z_2 = 1$ indicates $W = S_M$, while $Z_2 = 0$ indicates $W \neq S_M$; $Z_3 = 1$

indicates $W = S_L$, while $Z_3 = 0$ indicates $W \neq S_L$. Clearly, $Z_1 = Z_2 = Z_3 = 0$ indicate $W = S_N$. Note that no more than one indicator variable can be 1 simultaneously (i.e., W can only take one state at a time). In general, a qualitative variable that have m categories (states) need $m-1$ indicator variables to represent it.

3.2. General regression model

Let Y and $X_1, X_2, \cdots, X_n$ be the response variable and (quantitative) explanatory variables in a regression model, respectively. Let a qualitative variable W with m states (categories) be represented by indicator variables $Z_1, Z_2, \cdots, Z_{m-1}$. The qualitative variable can influence the regression model in the following four different ways (see Table 2):

Type	Regression Equation
Coincident:	$Y = B_0^0 + \sum_{i=1}^{n} B_i^0 X_i$
Parallel:	$Y = (B_0^0 + \sum_{j=1}^{m-1} B_0^j Z_j) + \sum_{i=1}^{n} B_i^0 X_i$
Concurrent:	$Y = B_0^0 + \sum_{i=1}^{n} (B_i^0 + \sum_{j=1}^{m-1} B_i^j Z_j) X_i$
General:	$Y = \underbrace{(B_0^0 + \sum_{j=1}^{m-1} B_0^j Z_j)}_{intercepts} + \sum_{i=1}^{n} \underbrace{(B_i^0 + \sum_{j=1}^{m-1} B_i^j Z_j)}_{slopes} X_i$

Table 2. Qualitative Regression Equation Forms

- *Coincident.* The relationship between the response and explanatory variables stays the same for all states of W. In other words, the equations for all states are coincident. This in fact is the situation for a static system environment assumed by the static query sampling method.
- *Parallel.* The relationship between the response and explanatory variables may differ in the intercept term but not the slope terms for different states of W. The relevant equation in Table 2 shows that the intercept term for the jth state of the qualitative variable is $B_0^0 + B_0^j$ ($j = 1, 2, \cdots, m$; and $B_0^m = 0$). Since the slope terms remain the same for all states, the equations for different states are parallel.
- *Concurrent.* The relationship between the response and explanatory variables may differ in the slope terms but not the intercept term for different states of W. The relevant equation in Table 2 shows that the ith slope term ($i = 1, 2, \cdots, n$) for the jth state of the qualitative variable is $B_i^0 + B_i^j$ ($j = 1, 2, \cdots, m$; and $B_i^m = 0$). The equations for different states have the same intercept term. They are said to be concurrent.
- *General.* The relationship between the response and explanatory variables may differ in both the intercept term and the slope terms for different states of the qualitative variable. This is the most general case.

Note that the cost of a query usually consists of (1) initialization cost such as moving a disk head to the right position; (2) I/O cost such as fetching a tuple from an operand table; and (3) CPU cost such as evaluating the qualification condition for a given tuple. A typical cost model for a unary query class may look like:

$$Y = B_0 + B_1 * N_U + B_2 * RN_U, \qquad (1)$$

where N_U and RN_U are the cardinalities of the operand table and result table, respectively; B_0, B_1 and B_2 are the parameters representing the initialization cost, the cost of retrieving a tuple from the operand table, and the cost of processing a tuple in the result table, respectively. Both B_1 and B_2 may reflect I/O as well as CPU costs. Therefore, the initialization cost affects the intercept term in a cost model, while the I/O and CPU costs affect the slope terms in the cost model. Clearly, the contention level of a system can significantly affect not only the initialization cost but also the I/O and CPU costs of a query because the resources like the disk, I/O bandwidth and CPU are shared by multiple processes. As a result, both the intercept and slope terms in a query cost model may change when the system contention level changes. Therefore, to incorporate a qualitative variable representing the system contention states into a query cost model, the general qualitative regression model is more appropriate.

3.3. Determining system contention states

Combining multiple dynamic environmental factors into a composite qualitative variable with a number of discrete contention states greatly simplifies the development of a cost model for a dynamic environment. The question now is how to determine an appropriate set of system contention states for a dynamic environment.

Two extremes

There are two extremes in determining a set of contention states. One extreme is to consider only one contention state for the system environment. A cost model developed in such a case is useful if the system environment is static. This, in fact, was the case that the static query sampling method assumed. However, as pointed out before, a real system environment may change dynamically over time. Using one contention state is obviously insufficient to describe the dynamic environment. For a dynamic environment, usually, the more the contention states are considered, the better a cost model. In principle, as long as we consider a sufficient number of contention states for the environment, we can get a satisfactory cost model. Another extreme is to consider an infinite number of contention states. However, the more the contention states are considered, the more the indicator variables are used in the cost model. The number of coefficients that need to be determined in a cost model therefore increases. Hence, if too many contention states are considered, the cost model can be very complicated, which is not good for either the development or maintenance of the cost model. In practice, as we will see in Section 5, a small number of contention states (three to six) are usually sufficient to yield a good cost model.

Determining states via iterative uniform partition

Notice that, for a given query, its cost increases as the system contention level increases (see Figure 1). Based on this observation, we can use the cost of a probing query to gauge the system contention level[2]. The range of probing costs (therefore, the contention level) is divided into subranges, each of which represents a contention state for the dynamic environment.

Let the cost C_{Q_p} of probing query Q_p fall in the range $[C_{min},\ C_{max}]$ in a dynamic environment. A simple way to determine the system contention states is to partition range $[C_{min},\ C_{max}]$ into subranges with an equal size. In other words, to determine m contention states[3] $S_m, S_{m-1}, \cdots, S_1$, we divide range $[C_{min},\ C_{max}]$ into m subranges $I_i = [C_{min}+(m-i)*D,\ C_{min}+(m-i+1)*D)$ and $I_1 = [C_{min} + (m-1) * D,\ C_{min} + m * D]$ where $i = m, m-1, ..., 2$ and $D = (C_{max} - C_{min})/m$. The system environment is said to be in contention state S_i if $C_{Q_p} \in I_i$ $(i = m, m-1, ..., 1)$. To obtain more system contention states, we can simply increase m. Hence, $\{\ I_m,\ I_{m-1},\ ...,\ I_1\ \}$ yields a set Γ_m of the system contention states for the dynamic environment.

Using this partition, it is easy to determine the system contention state in which a query is executed. Let $SP = \{\ Q_j \mid j = 1, 2, ..., s\ \}$ be a set of sample queries which are performed in a dynamic environment and whose observed data (costs, result table sizes, etc.) are to be used to derive a regression cost model for a query class. To determine the system contention state S_{Q_j} in which Q_j is executed, the cost C_{Q_p} of probing query Q_p in the same environment is measured. $S_{Q_j} = S_i$ if $C_Q \in I_i$ $(1 \leq i \leq m)$. We call the costs of a probing query associated with the sample queries are sampled probing query costs.

One basic question is how to determine a proper m. Another question is how to eliminate some unnecessary separations of subranges. Clearly, if the performance behaviors of queries in contention states S_{j-1} and S_j (for some j) are similar, separating S_{j-1} and S_j is unnecessary. The determination of system contention states should balance the accuracy and simplicity (hence low maintenance overhead) of a derived cost model.

[2]Our experiments showed that most queries, except the ones with extremely small cost (e.g., several hundredths of a second), can well serve as a probing query to gauge the system contention level.

[3]A decreasing index is used here to simplify the descriptions of the algorithms and derived cost models.

To solve these two problems, the following algorithm is used to improve the above straightforward uniform partition:

ALGORITHM 3.1 : Contention States Determination via Iterative Uniform Partition with Merging Adjustment (*IUPMA*)

Input: Observed data of sample queries and their associated probing query costs
Output: A set of system contention states[4]
Method:
1. **begin**
2. Derive a qualitative regression model with one contention state using the sample query data;
3. Let R^2_{new} be the coefficient of total determination of the current regression model;
4. Let s_{new} be the standard error of estimation of the current regression model;
5. $m := 1$;
6. **do**
7. $R^2_{old} := R^2_{new}$; $s_{old} := s_{new}$
8. $m := m + 1$;
9. Obtain a set Γ_m of m contention states for the system environment via the straightforward uniform partition;
10. Derive a qualitative regression model with m contention states using sample query data;
11. Let R^2_{new} be the coefficient of total determination for the current regression model;
12. Let s_{new} be the standard error of estimation of the current regression model;
13. **until** $(|(R^2_{new} - R^2_{old})/R^2_{old}|$ and $|(s_{new} - s_{old})/s_{old}|)$ are sufficiently small or m is too large;
14. $m := m - 1$;
15. Let S_j $(j = m, m-1, ..., 1)$ represent the current m contention states in Γ_m;
16. Let $A_i^j = B_i^0 + B_i^j$ $(i = 0, 1, ..., n)$ be the adjusted coefficient of ith variable X_i for state S_j in the general model in Table 2, where $X_0 \equiv 1$ is a dummy variable for the intercept term;
17. **for** $k = m$ **down to** 2 **do**
18. $E_k := \max_{i \in \{0,1,2,...,n\}} \{\ |(A_i^{k-1} - A_i^k)/A_i^k|\ \}$
19. **if** E_k is too small **then**
20. tag that states S_k and S_{k-1} should be merged;
21. **end for**
22. **if** some states are tagged to be merged **then**
23. Derive a qualitative regression model with new merged states using sample query data;
24. **goto** step 15;
25. **end if**;
26. **return** the current set Γ_m of contention states;
27. **end.**

There are two phases in Algorithm 3.1. The first phase is to determine a set of contention states via the uniform partition. The algorithm iteratively checks each qualitative regression model with an incremental number of contention states until (1) the model cannot be significantly improved in terms of the coefficient of total determination[5] R^2 and the standard error of estimation[6] s; or (2) too many contention states have been determined. Condition (2) is used here to prevent that a derived cost model becomes too complicated (in terms of the number of variables involved). The set of contention states obtained from the first phase are based on

[4]In fact, the algorithm integrates the contention states determination procedure with the cost model development procedure (to be discussed in the next section). As a result, a cost model is also produced as an output of the algorithm.

[5]The coefficient of total determination measures the proportion of variability in the response variable explained by the explanatory variables in a regression model [12]. The higher, the better.

[6]The standard error of estimation is an indication of the accuracy of estimation given by the model [12]. The smaller, the better

the uniform partition of the probing query cost range (see Figure 2). The partition does not consider whether two states actually have significantly different effects on the cost model or not. It is possible that some neighboring states have only slight different effects on the cost model. If so, the states should be merged into one to simplify the cost model. Such a merging adjustment is done during the second phase of the algorithm. If the maximum of relative errors of the corresponding pairs of adjusted coefficients (i.e., $B_i^0 + B_i^k$, and $B_i^0 + B_i^{k-1}$, $i = 0, 1, ..., n$) for two states S_k and S_{k-1} is too small, these two states are considered not to have significantly different effects on the cost model. The subranges in the final adjusted partition of probing query cost range may not have an equal size.

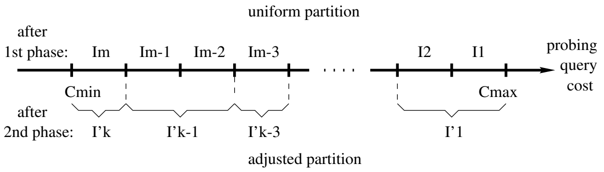

Figure 2. Contention States Determination via IUPMA

Determining states via data clustering

To capture the effect of every contention level on query costs for a dynamic environment in a cost model, we can let each contention level point have an equal chance to be chosen for running a given sample query. In other words, the probing query costs associated with the sample queries to indicate the sampled contention level points follow the uniform distribution within their range. A cost model derived by using such sample data can be used to estimate the cost of a query executed at any contention level. However, in a real dynamic application environment, the contention level may occur more often in some subranges than the others. To better capture the performance behavior of a dynamic environment, we can choose the contention level points for running sample queries based on the actual distribution of the contention level in the dynamic environment. As a result, the associated probing query costs may not follow the uniform distribution in their range. More often they are grouped into clusters.

Although Algorithm 3.1 is designed for uniformly distributed probing query costs, it usually can also handle clustered probing query costs well due to its iterating and adjusting mechanisms. However, the resulting partition of the probing query cost range for the clustered cases may not be the best since the boundaries considered at each iteration in the algorithm are fixed, regardless of the distribution of the system contention level. To overcome the problem, a data mining algorithm for data clustering can be incorporated into the contention states determination procedure here.

An agglomerative hierarchical algorithm is often used for data clustering[6]. The main idea of the algorithm is to place each data object in its own cluster initially and then gradually merge clusters into larger and larger clusters until a desired number of clusters have been found. The criterion used to merge two clusters ω_1 and ω_2 is to make their distance minimized. One widely used distance measure is the distance between the centroids or means $m(\omega_1)$ and $m(\omega_2)$ of two clusters, i.e., $D_{mean}(\omega_1, \omega_2) = ||m(\omega_1) - m(\omega_2)||$.

Let K be the maximum allowed number of system contention states. The above clustering algorithm can be used to obtain clusterings $\Omega_m = \{ \omega_m, \omega_{m-1}, ..., \omega_1 \}$ ($m = K, K-2, ..., 1$; ω_i's are clusters such that $m(\omega_{i+1}) < m(\omega_i)$ for $i = m-1, m-2, ..., 1$) for sampled probing query costs. Let subranges $I_j = [a_{j+1}, a_j)$ and $I_1 = [a_2, a_1]$, where $j = m, m-1, ..., 2$ and $a_{m+1} = C_{min}$, $a_m = (min(\omega_m) - max(\omega_{m-1}))/2$, ..., $a_2 = (min(\omega_2) - max(\omega_1))/2$, $a_1 = C_{max}$, here $min(\omega_i)$ and $max(\omega_i)$ are the minimum and maximum probing query costs in cluster ω_i. Clearly, $\{ I_m, I_{m-1}, ..., I_1 \}$ gives a set Γ_m of the system contention states for the dynamic environment, which reflects the distribution information of probing query costs (the contention level). If we use such Γ_m in Line 9 in Algorithm 3.1, we get a new algorithm, termed as the *Contention States Determination via Iterative Clustering with Merging Adjustment (ICMA)*.

Note that, for clustered probing query costs, it is possible that a cluster may not have a sufficient number of sampled data points to meet the minimum requirement for regression analysis. In such a case, we draw additional sample data points (via executing more sample queries) to make the cluster meet the minimum requirement rather than simply treat the data points in the cluster as outliers and ignore them. Although this way may change the distribution of the contention level sightly, no useful contention level points are ignored in the derived cost model.

Probing costs estimation

To minimize the overhead for determining a system contention state, a query with a small cost is preferred as a probing query. To further reduce the overhead, estimated costs (rather than observed costs) of probing query Q_p can be used to determine the contention states of a dynamic environment. The idea is to first develop a regression equation between the probing query cost Y_{Q_p} and some major system contention parameters[7] (such as CPU load ld_1, I/O utilization io, and size of used memory space um for a dynamic environment in Table 1), i.e.,

$$Y_{Q_p} = E_0 + E_1 * ld_1 + E_2 * io + E_3 * um, \tag{2}$$

[7] A standard statistical procedure can be used to determine the significant parameters for a system environment.

where E_i ($i = 0, 1, 2, 3$) are regression coefficients. Afterwards, every time when we want to determine the system contention state in which a query is executed we only need to check which subrange the estimated cost Y_{Q_p} of probing query Q_p lies in by using (2) without actually executing the probing query. Since obtaining the parameter values (ld_1, io, um) in (2) usually requires less overhead than executing a probing query, using the estimated costs of a probing query to determine system contention states is usually more efficient. However, estimation errors may introduce certain inaccuracy.

4. Development of cost models

As mentioned before, we extend the query sampling method for a static environment in [17] so as to develop cost models for a dynamic environment via introducing a qualitative variable. Such extensions are discussed in this section.

4.1. Query classification and sampling

Similar to the static query sampling method, we group local queries on a local database system into classes based on their potential access methods to be employed. The previous classification rules and procedures in [17] can be utilized. For example,

$$G_{1\,1} = \{\, \pi_\alpha(\sigma_F(R)) \mid \alpha\ is\ a\ list\ of\ columns\ in\ table\ R\ and\ qualification\ F\ has\ at\ least\ one\ conjunct\ R.a = C,\ where\ R.a\ is\ a\ clustered-indexed\ column,\ \pi\ and\ \sigma\ are\ project\ and\ select\ opreations\ respectively \}$$

is a class of unary queries that are most likely performed by using a clustered-index scan access method in a DBMS. Hence a similar performance behavior is shared among the queries in the class and can be described by a common cost model.

A sample of queries are then drawn from each query class in a similar way as before. However, since more parameters associated with the indicator variables are included in a cost model, more sample queries need to be drawn in order to meet the commonly-used rule for sampling in statistics, i.e., sample at least 10 observations for every parameter to be estimated [12]. The following proposition gives a guideline on the minimum number of sample queries needed for regression analysis.

PROPOSITION 4.1 *For the general qualitative regression cost model in Table 2 with n quantitative explanatory variables and one qualitative variable for m states, at least $10 * (m * (n + 1) + 1)$ observations need to be sampled.*

PROOF. Notice that there are $(n + 1)$ groups of regression coefficients in the cost model, one for each independent quantitative variable plus the intercept term. Each group has m coefficients, one for each state of the qualitative variable. In addition, the variance of error terms need also to be estimated. ∎

Sample queries drawn from a query class are performed in a dynamic environment. Their observed data as well as their associated probing query costs are recorded and used to derive a regression cost model for the query class. A load builder, which is part of the MDBS agent for each local DBS [2], is used to simulate a dynamic application environment at a local site in an MDBS during the query sampling procedure. The MDBS agent may also have an environment monitor which collects system statistics used for estimating the probing query costs when the estimation approach in Section 3.3 is employed.

4.2. Regression cost models

A qualitative regression cost model contains a set V_Q of quantitative explanatory variables and a set V_D of indicator variables for a qualitative variable indicating system contention states. Similar to the static query sampling method, we divide the cost model into two parts: $basic\ model{+}secondary\ part$. The basic model represents the essential part of the model, while the secondary part is used to further improve the model. The qualitative variable (i.e., the indicator variables) is included in both parts of the cost model to capture the dynamic environmental factors. Set V_Q is split into two subsets V_B and V_S, where V_B contains basic (quantitative) explanatory variables in the basic model, while V_S contains secondary (quantitative) explanatory variables in the secondary part. Table 3 lists potential explanatory variables in each of the subsets for a unary query class and a join query class. If all variables (including indicator variables) are included, the full cost model is:

$$Y = \underbrace{[(B_0^0 + \sum_{j=1}^{m-1} B_0^j * Z_j) + \sum_{X \in V_B} (B_X^0 + \sum_{j=1}^{m-1} B_X^j * Z_j) * X]}_{basic\ model} + \underbrace{[\sum_{X \in V_S} (B_X^0 + \sum_{j=1}^{m-1} B_X^j * Z_j) * X]}_{secondary\ part}$$

However, usually, not all variables are necessary for a given cost model.

To determine the variables to be included in a regression cost model for a query class, a mixed backward and forward procedure described below is adopted. We start with the full basic model which includes all variables in V_B and use a backward procedure to eliminate insignificant basic explanatory variables one by one. Note that, in our algorithm, if an explanatory variable X is removed from the

Class	Basic Explanatory Variables	Secondary Explanatory Variables
Unary Query Class	N_U – size (cardinality) of operand table TN_U – size of intermediate table RN_U – size of result table	L_U – tuple length of operand table RL_U – tuple length of result table NZ_U – operand table length $N_U * L_U$ RZ_U – result table length $RN_U * RL_U$
Join Query Class	N_{J1} – size of 1st operand table N_{J2} – size of 2nd operand table TN_{J1} – size of 1st intermediate table TN_{J2} – size of 2nd intermediate table RN_J – size of result table TN_{J12} – size of Cartesian product of intermediate tables	L_{J1} – tuple length of 1st operand table L_{J2} – tuple length of 2nd operand table RL_J – tuple length of result table NZ_{J1} – 1st operand table length $N_{J1} * L_{J1}$ NZ_{J2} – 2nd operand table length $N_{J2} * L_{J2}$ RZ_J – result table length $RN_J * RL_J$

Table 3. Potential Explanatory Variables for Cost Models

model, its coefficients $(B_X^0 + \sum_{j=1}^{m-1} B_X^j * Z_j)$ for all contention states (determined by indicator variables Z_j's) are removed. We then use a forward selection procedure to add more significant secondary explanatory variables from V_S into the cost model. This procedure tries to further improve the cost model. Similar to the backward procedure, if a secondary variable X is added into the model, its coefficients $(B_X^0 + \sum_{j=1}^{m-1} B_X^j * Z_j)$ for all contention states are included. Since it is expected that most basic variables are important to a cost model and only a few secondary explanatory variables are important, both the backward elimination and the forward selection procedures most likely terminate soon after they start.

Assume that we have n_j sampling observations in contention state S_j ($1 \le j \le m$), with $\sum_{j=1}^{m} n_j$ observations in total. Consider the simple correction coefficient between variables X and Y in contention state S_j:

$$r_{X.Y}^j = [\sum_{i=1}^{n_j} X_{ij} Y_{ij} - (\sum_{i=1}^{n_j} X_{ij} \sum_{i=1}^{n_j} Y_{ij})/n_j] / \sqrt{(\sum_{i=1}^{n_j} X_{ij}^2 - (\sum_{i=1}^{n_j} X_{ij})^2/n_j)(\sum_{i=1}^{n_j} Y_{ij}^2 - (\sum_{i=1}^{n_j} Y_{ij})^2/n_j)}$$

where X_{ij}, Y_{ij} are the values from the ith sampling observation ($1 \le i \le n_j$) in state S_j. For any explanatory variable X, if its maximum simple correlation coefficient $\max_{1 \le j \le m} \{r_{X.Y}^j\}$ with response variable Y is too small, it has little linear relationship with Y in any state. Such explanatory variables should be removed from consideration.

In the backward elimination procedure, the next variable X to be removed from the current model is the one which satisfies two conditions (a) its average simple correlation coefficient $\bar{r}_{X.Y} = (\sum_{j=1}^{m} r_{X.Y}^j)/m$ with response variable Y for all contention states is the smallest among all explanatory variables in the current model; (b) it makes $s' \le s$ or $|(s - s')/s| < \varepsilon$, where s' is the standard error of estimation for the reduced model (i.e., with X removed) given by:

$$s' = \sqrt{\frac{\sum_{j=1}^{m} \sum_{i=1}^{n_j} (Y_{ij} - \widehat{Y'}_{ij})^2}{\sum_{j=1}^{m} n_j - m * (k+1)}} \tag{3}$$

here $Y_{ij}, \widehat{Y}_{ij}, k$ denote the observed query cost, estimated query cost given by the reduced model, and number of explanatory variables in the model, respectively; s is the standard error of estimation for the original model given by a formula similar to (3); ε is a given small positive constant. Since the average simple correlation coefficient $\bar{r}_{X.Y}$ indicates the degree of linear relationship between X and Y on average in all states, foregoing condition (a) selects an explanatory variable X that contributes the least (on average in all states) in explaining the response variable Y. Since the standard error of estimation is an indication of estimation accuracy, foregoing condition (b) ensures that removing variable X from the model improves the estimation accuracy or affects the model very little. Removing a variable that has a little effect on the model can reduce the complexity and maintenance overhead of the model.

In the forward selection procedure, the next variable X from V_S to be added into the current model is the one satisfies (a) its average simple correlation coefficient $\bar{r}_{X.Y_s} = (\sum_{j=1}^{m} r_{X.Y_s}^j)/m$ with the residuals Y_s of the current model for all states is the largest among all explanatory variables in the model; i.e., it can explain the most (on average for all states) about the variations that the current model cannot explain; and (b) it significantly improves the estimation accuracy, i.e., $s' < s$ and $|(s - s')/s| > \varepsilon$, where s', s denote the standard errors of estimation for the augmented model (i.e., with X included) and the original model, respectively; and ε is a given small positive constant.

Note that the exact number of explanatory variables in a cost model is determined after the above mixed backward and forward procedure is done. However, we need such information to determine the query sample size from Proposition 4.1 at the beginning of the cost model development. Since it is expected that most basic explanatory variables in V_B are selected and only a few secondary explanatory variables in V_S are used for a cost model, we expect the number of explanatory variables in a cost model usually not exceed $|V_B| + \lceil |V_S|/2 \rceil$. Based on experiments, the maximum number M of contention states for a dynamic environment in practice can also be estimated. Hence, a reasonable query sample size is:

$$10 * (M * (|V_B| + \lceil |V_S|/2 \rceil + 1) + 1), \tag{4}$$

from Proposition 4.1.

4.3. Measures for developing useful models

Multicollinearity occurs when explanatory variables are highly correlated among themselves. In such a case, the estimated regression coefficients tend to have large sampling variability. It is better to avoid multicollinearity.

The presence of multicollinearity is detected by means of the variance inflation factor VIF [11]. When an explanatory variable has a strong linear relationship with the other explanatory variables, its VIF is large. In a dynamic environment with multiple contention states, let VIF_j ($1 \le j \le m$) be the variance inflation factor of explanatory variable

Query Class	Cost Estimation Model with Qualitative Variable (i.e., Multi-States Cost Models)
$G_1/DB2$	$(-0.1522e{+}1-0.5784e{+}0*Z_2+0.1760e{+}1*Z_1)+(-0.1333e{-}4-0.2149e{-}3*Z_2+0.4738e{-}4*Z_1)*TN_U+(0.5467e{-}5+0.8293e{-}4*Z_2+0.5145e{-}4*Z_1)*N_U+(0.1378e{-}2+0.1469e{-}2*Z_2+0.3310e{-}2*Z_1)*RN_U+(0.2912e{-}7+0.1636e{-}4*Z_2+0.8896e{-}5*Z_1)*NZ_U$
$G_2/DB2$	$(0.6758e{+}1-0.4563e{+}1*Z_5-0.1311e{+}2*Z_4-0.3462e{+}1*Z_3-0.1198e{+}2*Z_2+0.3981e{+}1*Z_1)+(0.6701e{-}4-0.3545e{-}4*Z_5+0.6882e{-}4*Z_4+0.1225e{-}3*Z_3-0.1153e{-}3*Z_2+0.1855e{-}3*Z_1)*N_U+(0.6153e{-}3+0.7202e{-}3*Z_5+0.1472e{-}2*Z_4+0.2740e{-}2*Z_3+0.3729e{-}2*Z_2+0.4015e{-}2*Z_1)*RN_U+(0.5499e{-}1+0.8548e{+}0*Z_5+0.8651e{+}0*Z_4+0.1126e{+}1*Z_3+0.2258e{+}1*Z_2+0.2269e{+}1*Z_1)*RL_U+(-0.1155e{+}1+0.7266e{+}0*Z_5+0.1288e{+}1*Z_4-0.3080e{+}0*Z_3+0.1167e{+}1*Z_2-0.1680e{+}1*Z_1)*L_U$
$G_3/DB2$	$(0.1232e{+}2+0.6065e{+}2*Z_2-0.3505e{+}2*Z_1)+(0.5634e{-}7+0.6310e{-}8*Z_2+0.3707e{-}6*Z_1)*TN_{J12}+(0.8489e{-}3+0.1586e{-}2*Z_2+0.3656e{-}2*Z_1)*RN_J$
$G_1/Oracle$	$(0.1648e{-}1-0.5209e{+}0*Z_2+0.2931e{+}1*Z_1)+(-0.3030e{-}3+0.1586e{-}3*Z_2+0.3281e{-}3*Z_1)*TN_U+(-0.1549e{-}4+0.4146e{-}4*Z_2+0.6483e{-}4*Z_1)*N_U+(0.2691e{-}2+0.1986e{-}2*Z_2+0.3791e{-}2*Z_1)*RN_U+(-0.5699e{-}4-0.1221e{-}4*Z_2+0.8089e{-}4*Z_1)*RZ_U+(0.8557e{-}5+0.3322e{-}5*Z_2+0.1894e{-}6*Z_1)*NZ_U$
$G_2/Oracle$	$(0.5262e{+}1-0.7762e{+}1*Z_5-0.3278e{+}1*Z_4-0.7294e{+}1*Z_3-0.7848e{+}1*Z_2-0.2817e{+}1*Z_1)+(-0.1034e{-}3+0.1738e{-}3*Z_5+0.1651e{-}3*Z_4+0.2833e{-}3*Z_3+0.2526e{-}3*Z_2+0.2542e{-}3*Z_1)*N_U+(0.5224e{-}3+0.2012e{-}2*Z_5+0.2955e{-}2*Z_4+0.4490e{-}2*Z_3+0.6067e{-}2*Z_2+0.6541e{-}2*Z_1)*RN_U$
$G_3/Oracle$	$(-0.1457e+2-0.4381e{+}2*Z_3+0.7830e{+}2*Z_2-0.7726e{+}2*Z_1)+(-0.9777e{-}7+0.1322e{-}6*Z_3+0.2320e{-}6*Z_2+0.1373e{-}6*Z_1)*TN_{J12}+(0.1257e{-}2+0.1737e{-}2*Z_3+0.3801e{-}2*Z_2+0.5704e{-}2*Z_1)*RN_J+(0.7793e{-}3-0.4290e{-}3*Z_3-0.8994e{-}3*Z_2-0.1771e{-}6*Z_1)*TN_{J1}+(-0.1887e{+}1+0.4723e{+}1*Z_3+0.8677e{+}1*Z_2+0.9416e{+}1*Z_1)*RL_J+(0.6537e{+}1+0.2638e{-}1*Z_3-0.2248e{+}2*Z_2+0.3179e{+}1*Z_1)*L_{J1}$

Table 4. Multi-State Cost Models for DB2 and Oracle

X in state S_j. If $\min_{1\le j\le m}\{VIF_j\}$ is large, X is not included in a cost model to avoid multicollinearity.

F-test, the standard error of estimation s, the coefficient of multiple determination R^2, as well as the percentage of good cost estimates for test queries are used to validate the significance of a developed regression cost model.

5. Experimental results

To verify the feasibility of our multi-states query sampling method for developing cost models in a dynamic environment, experiments were conducted in a multidatabase environment using a research prototype called CORDS-MDBS [2]. Two commercial DBMSs, i.e., Oracle v8.0 and DB2 v5.0, were used as local database systems running under Solaris 5.1 on two SUN UltraSparc 2 workstations. Figure 3 shows the experimental environment. Local queries are submitted to a local DBS via an MDBS agent. The MDBS agent provides a uniform relational ODBC interface for the global server. It also contains a load builder which generates dynamic loads to simulate dynamic application environments.

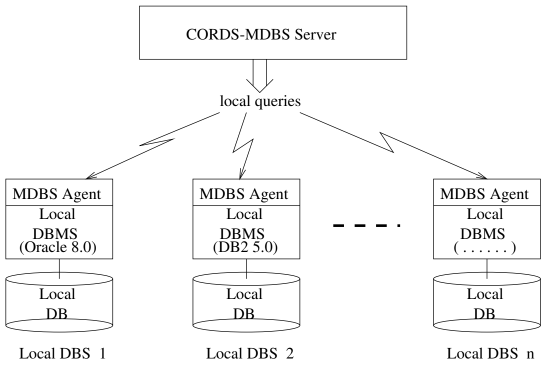

Figure 3. Experimental Environment

The local databases used in the experiments were the same as those in [17, 19], except that each table is ten-time larger than before due to the improved space availability and CPU capability in our experimental environment. More specifically, each local database has 12 randomly-generated tables $R_i(a_1, a_2, ..., a_j)$ $(i = 1, 2, ..., 12; j \in \{3, 5, 7, 9, 11, 13\})$ with cardinalities ranging from 3,000 $\sim$ 250,000. Each table has a number of indexed columns and various selectivities for different columns.

In the experiments, queries on each local DBS were classified first according to the same rules in the static query sampling method. A sample of queries with the size meeting condition (4) were then drawn from each query class and performed in the simulated dynamic environments at the local sites. Their observed costs together with the associated probing query costs are used to derive a cost model with a qualitative variable for each query class using the techniques suggested in the previous sections. Some randomly-generated test queries [17] in the relevant query classes were also performed in the dynamic environment, and their observed costs were compared with the estimated costs given by the derived cost models. Note that, unlike the scientific computation in engineering, the accuracy of cost estimation in query optimization is not required to be very high. The estimated costs with relative errors within 30% are considered to be very good, and the estimated costs that are within the range of one-time larger or smaller than the corresponding observed costs (e.g., 2 minutes vs. 4 minutes) are considered to be good. Only those estimated costs which are not of the same order of magnitude with the observed costs (e.g., 2 minutes vs. 3 hours) are not acceptable.

Table 4 shows the cost models derived by applying the multi-states query sampling method suggested in this paper for three representative query classes for each local DBS, namely[8], a unary query class G_1 without usable indexes, a unary query class G_2 with usable non-clustered indexes for ranges, and a join query class G_3 without usable indexes. Table 5 shows some statistical measures for the derived cost models[9]. For the comparison purpose, two static experimental cases were also considered. In the first case, cost models were derived by applying the static query sampling method to sampling data obtained from a *static* environment (Static Approach 1). In the second static case,

[8]The three query classes correspond to $G_{1\ 4}$, $G_{1\ 5}$, and $G_{2\ 4}$ in [17].
[9]The number in parentheses beside 'multi-states' in Table 5 indicates the number of contention states used for the relevant cost model.

cost models were derived by applying the static query sampling method to sampling data obtained from a *dynamic* environment (Static Approach 2). This in fact is to restrict the multi-states query sampling method to consider only one contention state.

query class	cost model type	R^2	s	average cost	very good estimates	good estimates
G_1 for $DB2$	multi-states (3)	0.972	0.157e+2	0.528e+2	55%	78%
	one-state	0.798	0.363e+2	0.511e+2	30%	58%
	static	0.972	0.672e+0	0.290e+1	3%	5%
G_2 for $DB2$	multi-states (6)	0.994	0.997e+1	0.620e+2	60%	76%
	one-state	0.779	0.620e+2	0.690e+2	24%	48%
	static	0.986	0.733e+0	0.359e+1	7%	14%
G_3 for $DB2$	multi-states (3)	0.996	0.230e+3	0.735e+3	37%	62%
	one-state	0.910	0.254e+3	0.431e+3	27%	45%
	static	0.992	0.116e+2	0.381e+2	9%	13%
G_1 for $Oracle$	multi-states (3)	0.982	0.160e+2	0.680e+2	69%	81%
	one-state	0.876	0.576e+2	0.865e+2	35%	60%
	static	0.999	0.917e-1	0.402e+1	3%	6%
G_2 for $Oracle$	multi-states (6)	0.993	0.143e+2	0.873e+2	63%	74%
	one-state	0.901	0.672e+2	0.108e+3	35%	62%
	static	0.999	0.301e+0	0.493e+1	4%	8%
G_3 for $Oracle$	multi-states (4)	0.999	0.148e+3	0.998e+3	51%	67%
	one-state	0.951	0.507e+3	0.882e+3	22%	44%
	static	0.999	0.503e+1	0.492e+2	0%	1%

Table 5. Statistics for Cost Models

From the experimental results, we can have the following observations:

- The multi-states query sampling method presented in this paper can derive good cost models in a dynamic environment. The coefficients of total determination in Table 5 indicate that all derived models can capture 98.9% variations in query cost on average. The standard errors of estimation are acceptable, compared with the magnitude of the average cost of relevant sample queries (only 22% of average costs on average). The statistical F-tests at significance level $\alpha = 0.01$ were also conducted, which showed that all cost models are useful for estimating query costs in a dynamic environment.

- The (static) cost models derived by the static query sampling method for a static environment (i.e., Static Approach 1) are not suitable for estimating query costs in a dynamic environment. Although such cost models may have good coefficients of total determination (99.1% on average in Table 5) for the sampling data in a static environment, they can hardly give good cost estimates in a dynamic environment (gave only 7.8% good cost estimates on average in Table 5 for the test queries in our experiments).

- The (multi-states) cost models derived by using the multi-states query sampling method for a dynamic environment significantly improve the (one-state) cost models derived by applying the static query sampling method for the dynamic environment (i.e., Static Approach 2). In fact, compared with the one-state cost models, the multi-states cost models increase the number of very good cost estimates (i.e., with relative errors ≤ 0.3) and the number of good cost estimates (i.e., within one time range) by 27.0% and 20.2% (on average) respectively for the test queries. Figures 6 $\sim$ 5 show comparisons among the observed costs, estimated costs by the multi-states cost models, and estimated costs by the one-state cost models for the test queries in a dynamic environment.

- The more contention states are considered, the better the derived cost model usually is. For example, the coefficients of total determination for the cost models for query class $G_2/Oracle$ with 1 to 6 contention states are 0.7788, 0.9636, 0.9674, 0.9899, 0.9922, respectively. However, the improvement may be very small after the number of contention states reaches certain point. Table 5 shows that usually considering 3 to 6 contention states for a dynamic environment is sufficient to obtain a good cost model.

- Like static techniques [3, 17], it is also true to the multi-states query sampling method that small-cost queries usually have worse cost estimates than large-cost queries. The main reason for this is that even a small momentary change in the system environment may have a significant impact on the cost of a small-cost query. It is not easy to capture all such small environmental changes in a cost model. Fortunately, estimating the costs of small-cost queries is not as important as estimating the costs of large-cost queries because it is more important to identify large-cost queries so that inefficient execution plans can be avoided.

- Contention states determination algorithm *IUPMA* works well for both uniformly-distributed and clustered probing query costs, while algorithm *ICMA* can determine an even better set of system contention states for the clustered cases. Note that the sampled probing query costs were drawn by following the distribution of the contention level in a dynamic environment. In fact, the experimental results shown in Tables 4 $\sim$ 5 and Figures 4 $\sim$ 9 were obtained for the uniform case. Extensive experiments were also conducted for clustered cases. The experimental results showed that, for a given query class, the cost model derived in the clustered cases is usually better than the one derived for the uniform case even if *IUPMA* is used. This is because the cost models for the clustered cases only need to capture performance behavior of queries in more focused and narrower subrange(s) of the contention level. Table 6 shows some typical experimental results for a query class in a dynamic environment with clustered contention levels (see Figure 10 for the relevant frequency distribution of the contention level).

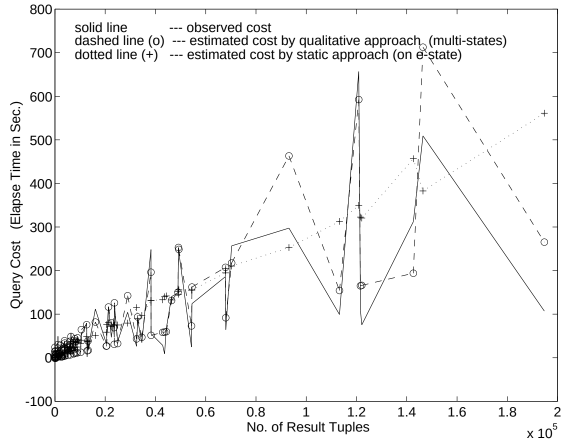

Figure 4. Costs for Test Queries in G_1 on DB2 5.0

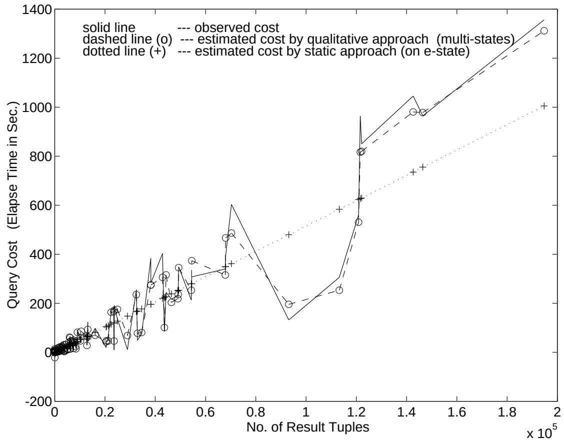

Figure 5. Costs for Test Queries in G_1 on Oracle 8.0

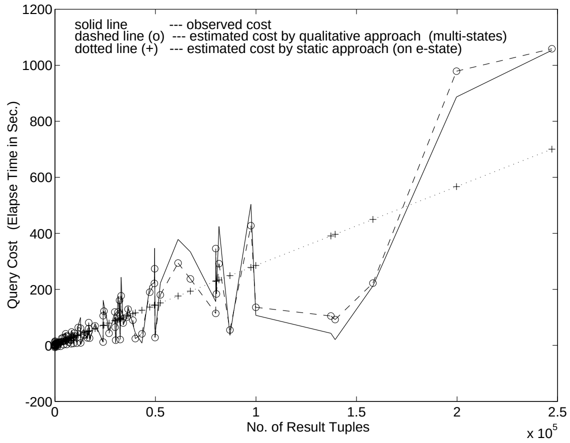

Figure 6. Costs for Test Queries in G_2 on DB2 5.0

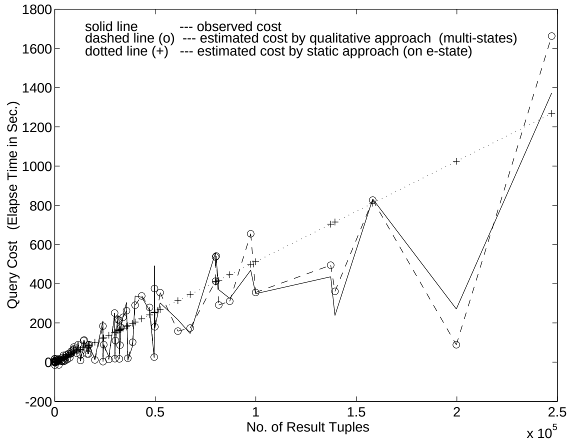

Figure 7. Costs for Test Queries in G_2 on Oracle 8.0

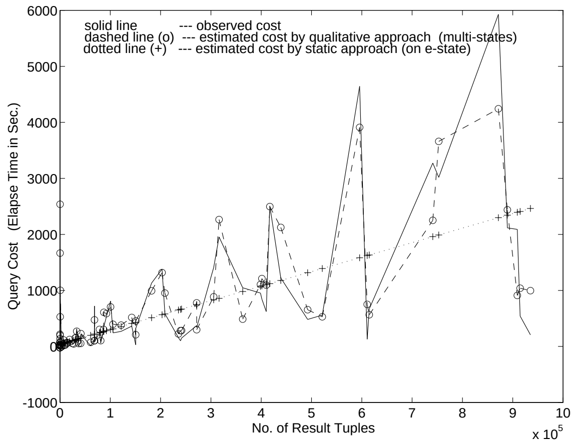

Figure 8. Costs for Test Queries in G_3 on DB2 5.0

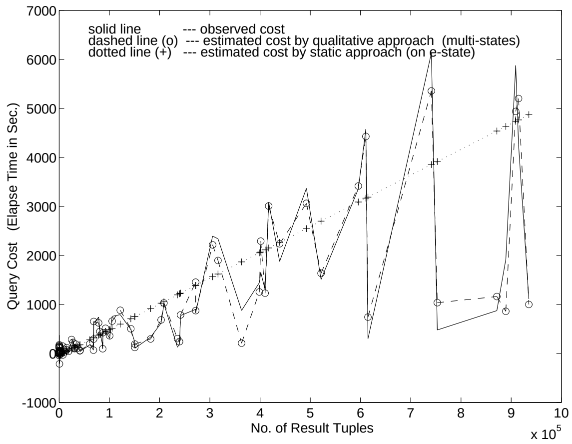

Figure 9. Costs for Test Queries in G_3 on Oracle 8.0

query class	states determination	# of states	R^2	s	average cost	very good estimates	good estimates
G_1 for	IUPMA	3	0.978	0.128e+2	0.488e+2	58%	82%
$DB2$	ICMA	3	0.991	0.740e+1	0.465e+2	82%	95%

Table 6. Statistics for Cost Models in a Clustered Case

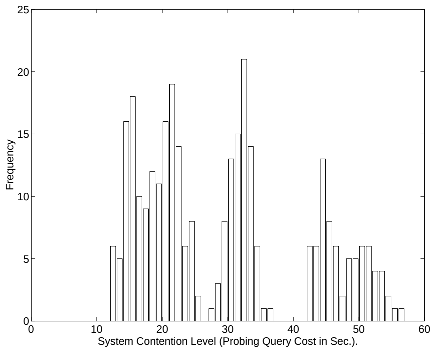

Figure 10. Histogram of Contention Level in a Clustered Case

6. Conclusions

The techniques proposed so far in the literature to develop local cost models in an MDBS are only suitable for a static environment. Many dynamically-changing environmental factors have significant effects on query cost. To develop a cost model for a dynamic environment, we have proposed a new qualitative approach, called the multi-states query sampling method, in this paper. This method solves the dynamic problem by dividing the system contention level, which reflects the combined net effect of dynamic factors on query cost, in a dynamic environment into a number of discrete contention states based on the costs of a probing query and then incorporating a qualitative variable indicating the contention states into a cost model. The costs of a probing query can be either observed or estimated. An appropriate set of system contention states can be determined based on either an iterative uniform partition with merging adjustment or a clustering-based partition. The former is designed for a dynamic environment with the contention level following the uniform distribution, while the latter is suitable for a dynamic environment with the contention level following a non-uniform distribution with clusters. Due to the iterating and adjusting mechanisms, the former usually can also handle the cases with non-uniform distributions although the latter may do a better job. The development of regression cost models for a dynamic environment is based on the extensions of techniques from our previous static query sampling method. Our experimental results demonstrate that the multi-states query sampling method presented in this paper is quite promising in developing useful cost models in a dynamic environment. It represents a significant improvement over the static techniques in a dynamic environment. Usually, considering a small number of contention states is sufficient to yield a good cost model.

Although dynamic environmental factors have significant effects on query cost, they were ignored in most existing cost models for MDBSs or other database systems due to lack of appropriate techniques. This paper introduces a promising approach to tackling the problem. However, further research needs to be done in order to fully solve all relevant issues.

References

[1] S. Adali *et al*. Query caching and optimization in distributed mediator systems. In *Proc. of SIGMOD*, pp 137–48, 1996.

[2] G.K. Attaluri, D.P. Bradshaw, N. Coburn, P.-Å. Larson, P. Martin, A. Silberschatz, J. Slonim, and Q. Zhu. The CORDS multidatabase project. *IBM Systems Journal*, 34(1):39–62, 1995.

[3] W. Du, *et al*. Query optimization in heterogeneous DBMS. In *Proc. of VLDB*, pp 277–91, 1992.

[4] W. Du, M. C. Shan, and U. Dayal. Reducing Multidatabase Query Response Time by Tree Balancing. In *Proc. of SIGMOD*, pp 293 – 303, 1995.

[5] G. Gardarin, *et al*. Calibrating the query optimizer cost model of IRO-DB, an object-oriented federated database system. In *Proc. of VLDB*, pp 378–89, 1996.

[6] S. Guha, *et al*. CURE: An Efficient Clustering Algorithm for Large Databases. In *Proc. of SIGMOD*, pp 73–84, 1998.

[7] C. Lee and C.-J. Chen. Query Optimization in Multidatabase Systems Considering Schema Conflicts. *IEEE Trans. on Knowledge and Data Eng.*, 9(6):941–55, 1997.

[8] W. Litwin, *et al*. Interoperability of multiple autonomous databases. *ACM Comp. Surveys*, 22(3):267–293, 1990.

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

[10] H. Naacke, G. Gardarin, and A. Tomasic. Leveraging mediator cost models with heterogeneous data sources. In *Proc. of 14th Int'l Conf. on Data Eng.*, pp 351–60, 1998.

[11] J. Neter, *et al*. *Applied Linear Statistical Models, 3rd Ed.* Richard D. Irwin, Inc., 1990.

[12] R. Pfaffenberger *et al*. *Statistical Methods for Business and Economics*. Richard D. Irwin, Inc., 1987.

[13] M. T. Roth, F. Ozcan, and L. M. Haas. Cost models DO matter: providing cost information for diverse data sources in a federated system. In *Proc. of VLDB*, pp 599–610, 1999.

[14] A. P. Sheth, *et al*. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, Sept. 1990.

[15] T. Urhan, *et al*. Cost-based query scrambling for initial delays. In *Proc. of SIGMOD.*, pp 130–41, 1998.

[16] Q. Zhu and P.-Å. Larson. A fuzzy query optimization approach for multidatabase systems. *Int'l J. of Uncertainty, Fuzziness and Knowledge-Based Sys.*, 5(6):701 – 22, 1997.

[17] Q. Zhu and P.-Å. Larson. Solving local cost estimation problem for global query optimization in multidatabase systems. *Distributed and Parallel Databases*, 6(4): 373 – 420, 1998.

[18] Q. Zhu and P.-Å. Larson. Building regression cost models for multidatabase systems. In *Proc. of 4th IEEE Int'l Conf. on Paral. and Distr. Inf. Syst.*, pp 220–31, Dec. 1996.

[19] Q. Zhu and P.-Å. Larson. A query sampling method for estimating local cost parameters in a multidatabase system. In *Proc. of 10th IEEE Int'l Conf. on Data Eng.*, pp 144–53, Feb. 1994.