Query Processing Issues in Image(multimedia) Databases

Surya Nepal and M.V. Ramakrishna
Department of Computer Science
RMIT University
GPO Box 2476V, Melbourne VIC 3001
{ nepal,rama}@cs.rmit.edu.au

Abstract

Multimedia databases have attracted academic and industrial interest, and systems such as QBIC (Content Based Image Retrieval system from IBM) have been released. Such systems are essential to effectively and efficiently use the existing large collections of image data in the modern computing environment. The aim of such systems is to enable retrieval of images based on their contents. This problem has brought together the (decades old) database and image processing communities.

As part of our research in this area, we are building a prototype CBIR system called CHITRA. This uses a four level data model, and we have defined a Fuzzy Object Query Language(FOQL) for this system. This system enables retrieval based on high level concepts, such as "retrieve images of MOUNTAINS", "retrieve images of MOUNTAINS and SUNSET".

A problem faced in this system is processing of complex queries such as "retrieve all images that have similar color histogram AND similar texture to the given example image". Such problems have attracted research attention in recent times, notably by Fagin, Chaudhury and Gravano. Fagin has given an algorithm for processing such queries and provided a probabilistic upper bound for the complexity of the algorithm (which has been implemented in IBM's Garlic project). In this paper we provide theoretical (probabilistic) analysis of the expected cost of this algorithm. We propose a new multi-step query processing algorithm and prove that it performs better than Fagin's algorithm in all cases. Our algorithm requires fewer database accesses. We have evaluated both algorithms against an image database of 1000 images on our CHITRA system. We have used color histogram and Gabor texture features. Our analysis presented and the reported experimental results validate our algorithm (which has significant performance improvement).

1. Introduction

Advances in storage and processing technologies have made multimedia databases (comprising large collections of images, video and sound) have become possible. Such databases will be as common in the next century as relational databases today. Storing, analyzing, transmitting and retrieving of multimedia objects has become an important research area. Images are widely used in multimedia applications such as in medical imaging, geographic information systems, and photo galleries [6]. Traditionally, image retrieval was based on manually entered keywords associated with them. A user can pose queries by typing keywords that represent his/her information need and the system retrieves images using indexes on keywords. This does not scale well, and users always want to retrieve images based on their contents rather than predefined keywords. This has led to a new paradigm called content based image retrieval (CBIR). This has brought together the database and image processing communities, and both commercial developers and academic research community are actively involved in this area. Recently there have been several CBIR systems developed as commercial/academic prototypes and some released into the market [6, 13].

These CBIR systems extract image features(usually high dimensional vectors) and index them using a structure such as R*-trees [1]. Users can pose queries based on example images, color pallet, texture patterns or sketches. The system ranks the images in the database based on some similarity measure and retrieves images that are most similar to the users' query [12]. Many algorithms and strategies have been proposed for similarity based retrieval from the index structures. However, there has been little work on optimizing user queries at the high level, exploring the underlying index structure [5, 3]. This paper addresses how to use this low level (well researched) mechanism to process higher level queries.

As a part of our research in this area, we are developing a CBIR prototype system called CHITRA [10]. It uses a four level data model we have developed to store the images, features extracted and semantic information [9]. The raw image data is stored at the lowest level. The features extracted (the user can specify new features and provide routines to compute them) are stored in the second level. The system defined and user provided similarity functions and spatial relationships are in the third level. The user defined semantics is at the highest level. The system enables users to define the high level concepts such as SUNSET, MOUNTAINS by interaction (giving example images etc.). The user can pose queries such as "retrieve all images that have SUNSET and MOUNTAIN" and "retrieve all images that have red color and fine texture". The topic of this paper, is efficient processing of complex queries such as "retrieve all images that have similar color histogram AND similar texture to the given example image".

A user query in a CBIR system may be simple dealing with a single feature (color, texture, etc.) or complex, dealing with a combination of features (such as color and texture, texture and shape). Evaluation of simple queries has been well researched, and can be accomplished efficiently using underlying high dimensional indexing structures. Many efficient algorithms/strategies have been proposed for range queries, k-nearest neighbor queries and its variants [1] and are implemented in many existing CBIR systems [6]. There has been little research for efficient evaluation of complex queries(such as above) using the underlying index structures. Recently Fagin [5], and Chaudhuri and Gravano [3] have addressed this problem. The required result of a query is a sorted list of images in the database based on some similarity measure. Both approaches have used fuzzy logic to arrive at the sorted list to evaluate and answer the complex queries. Fagin's algorithm has been implemented in IBM's Garlic project [4]. He has given a probabilistic upper bound for the complexity of the algorithm for evaluating the conjunctive queries. Chaudhuri and Gravano's strategy for query optimization is based on the notion of optimal execution space. The main focus of their work is on the choice of indexes for optimal execution of queries with threshold filter conditions (such as "retrieve all the images that are similar to example image where color similarity is greater than 0.9 and texture similarity is greater than 0.8"). Their results are not significant for CBIR systems such as our CHITRA. A user query seldom contains a threshold filter conditions. A user can not provide proper threshold condition without knowing the details of underlying similarity measures. A filter condition that does not have a threshold value or has a bad threshold value results in serial access (in the above query, the selectivity will be 1, if all the images in the database have texture similarity above 0.8).

Combining similarity measures has been well studied in text retrieval context [11]. Many combining algorithms have been proposed and evaluated. Our approach in CHITRA is based on Fagin's work. We refer to his algorithm as a single-step query processing algorithm. His probabilistic bounds of the algorithm is under the assumption of random distribution of similarity values (in the range of 0.0 to 1.0). In this paper, we first derive an expression for the expected cost of the algorithm under the same model. The practical significance of these results is dependent on the exact distribution of similarity values(which is unknown). To determine this significance, we have conducted experiments on our CHITRA and reported the results. Our analysis of the Fagin's algorithm, led to an observation that we can do better with a class of fuzzy combining functions(such as the standard min and max functions) which satisfy certain bounding properties. We propose a multi-step algorithm for query evaluation. Our analysis shows that multi-step algorithm performs better than single-step algorithm in all cases. We have experimentally evaluated the comparative performance of the two algorithms with a database of 1000 images on our CHITRA system using color histogram and Gabor texture features.

The rest of the paper is organized as follows. Section 2 describes statement of the problem and development of suitable framework. The theoretical and experimental analysis of single-step query processing algorithm is presented in section 3. The proposed multi-step query processing algorithm is described in section 4, along with its theoretical analysis. Experimental results of comparison are presented in section 5, and conclusions are presented in the last section.

2. Problem definition and framework development

We are mainly dealing with processing queries of the form *retrieve images whose (color = "red") AND (texture = "fine")*. Such queries are common in CBIR systems, where we need to retrieve images that are similar to the given feature vectors (where feature vectors are given by selecting example images or color pallet or texture patterns).

The CBIR systems use high dimensional indexing structures (of color and texture vectors for above example) to process such queries. The system only need to display the top k (say 10) images on the screen. The user can view next k images by pressing the next button available on the screen. Thus the system must provide an efficient evaluation of "next-k" request. The obvious naive algorithm to evaluate the query is as follows.

1. Find the graded set, based on color similarity, of all images and their grades(similarity with color vector).
2. Find the graded set, based on texture similarity, of all images and their grades(similarity with texture vector).

3. Combine the two graded sets using combining functions(to be defined) and sort them on resulting grade.

4. Display the top k images on the screen. When the user requests "next k", the system displays further k images from the sorted list.

This algorithm accesses color and texture feature vectors of all images from the database. The question is, how to process such queries optimally by using the underlying indexing structure? Thus our problem of query optimization is to retrieve the matching "next k" images from the database in an efficient manner.

The CBIR systems index each feature (referred to by a query term) of all images using multi-dimensional indexing structures such as R^*-tree [1]. The researchers of such index structures have studied various algorithms to process range query, and k-nearest neighbor query etc. against a given feature vector for these indexing structures. Thus we assume that a subsystem exists, which can yield a sorted list of images based on the similarity value to any given feature vector [5].

Definition 1 subsystem: *Each CBIR query specifies one or more feature vectors. For each such feature (e.g., color histogram) there exists an index of the corresponding feature vectors of all images in the database. Such index along with the similarity processing algorithms (e.g., k-nearest neighbor) is called a subsystem. Each subsystem returns images in a sorted order, of similarity with the given query feature vector.*

A graded set is a set of pairs (x, g), where x is the image and g (its grade) a real number in the interval $[0, 1]$. We regard the graded set returned by each subsystem as a fuzzy set. In CHITRA we make use of (fuzzy) combining functions to arrive at the final result (a graded set) using the graded sets returned by each subsystem. Researchers have investigated various rules(functions) for combining fuzzy boolean queries [5].

If x is an image and q a query term, let $\mu_q(x)$ denote the grade of x against the query q. We define t to be an $m-ary$ combining function which combines the result of m query subsystems.

$t : V^m \rightarrow V$, where $V = [0, 1]$.

We now consider combining functions to evaluate conjunctions of m query terms. Let $F(A_1, .., A_n)$ be an $m - ary$ query, where $A_1, .., A_n$ are query terms. We use $\mu_{F(A_1,..,A_m)}()$ to denote the semantics(grade) of $m - ary$ query $F(A_1, .., A_m)$.

Combining functions evaluating conjunctive and disjunctive queries must satisfy the the triangular norm and co-norm properties (of conservation, monotonicity, commutativity and associativity) [5].

Conservation: $t(0, 0) = 0; t(x, 1) = t(1, x) = x$ for norm and $t(1, 1) = 1; t(x, 0) = t(0, x) = x$ for co-norm.

Monotonicity: $t(x_1, x_2) \leq t(x_1\prime, x_2\prime)$ if $x_1 \leq x_1\prime$ and $x_2 \leq x_2\prime$.

Commutativity: $t(x_1, x_2) = t(x_2, x1)$

Associativity: $t(t(x_1, x_2)x_3) = t(x_1, t(x_2, x_3))$

Triangular norms and co-norms are dual. That is, if t is a norm, then $s(x, y) = 1 - t(1 - x, 1 - y)$ is a co-norm. Fagin has defined strictness property: $t(x_1, x_2, ..., x_m) = 1$ iff $x_i = 1$ for every i. We want the combining functions to satisfy two additional properties, lower and upper bounding properties defined below.

Definition 2 Lower bounding property: *A combining function t for the objects x and x' under the query $A_1 \wedge A_2 \wedge .... \wedge A_n$ satisfies the lower bounding property iff*

$\exists i \forall j : \mu_{A_i}(x) \leq \mu_{A_j}(x') \Rightarrow t(\mu_{A_1}(x), .., \mu_{A_m}(x)) \leq t((\mu_{A_1}(x'), .., \mu_{A_m}(x'))$

Definition 3 Upper bounding property: *A combining function s for the objects x and x' under the query $A_1 \vee A_2 \vee .... \vee A_n$ satisfies the upper bounding property iff*

$\exists i \forall j : \mu_{A_i}(x) \geq \mu_{A_j}(x') \Rightarrow s(\mu_{A_1}(x), .., \mu_{A_m}(x)) \geq s((\mu_{A_1}(x'), .., \mu_{A_m}(x'))$

The standard rules for fuzzy logic given below were defined by Zadeh.

Conjunction rule: $\mu_{q_1 \wedge q_2}$ = min$(\mu_{q_1}(x), \mu_{q_2}(x))$.

Disjunction rule: $\mu_{q_1 \vee q_2}$ = max$(\mu_{q_1}(x), \mu_{q_2}(x))$.

Negation rule: $\mu_{\neg q_1}(x) = 1 - \mu_{q_1}(x)$.

There are different combining functions available under the framework of fuzzy logic. The above rules are currently used in CHITRA. The standard fuzzy rule "min" satisfies the lower bounding property. The standard fuzzy rule "max" satisfies the upper bounding property. There are other functions that satisfy the lower and upper bounding properties. In this paper, all algorithms are evaluated using combining functions that satisfy the above bounding properties.

3. Fagin's Single-Step Query Processing Algorithm

Fagin proposed an algorithm, used in Garlic project, for combining fuzzy information from multiple repositories [5, 4]. We refer to this algorithm, given below, as single-step query processing algorithm. It can be used in CBIR systems, that support a subsystem as per definition 1, for each of the features involved. Both conjunctive and disjunctive queries can be evaluated using the algorithm. The algorithm returns the top(most similar) k images for a query $Q = F_t(A_1, .., A_m)$ using the combining function t. It has three phases: sorted access, random access and computation.

1. The subsystem i processes the query term A_i. For each i the corresponding subsystem outputs pairs $(x, \mu_{A_i}(x))$ one by one in sorted order of $\mu_{A_i}(x)$ (the similarity of image x under A_i).

   The above step is repeated until there are at least "k-matches". That is, repeat until there is a set L of at least k images such that each subsystem i has output all of the members of L. Let X_T^i be the first T images output by i. Then the step 1 is repeated until $L = \cap_i^m X_T^i$ contains at least k members.

2. For each image x that has been "visited" (for each $x \in X_T^i$), do a "random access" in each subsystem j to find $\mu_{A_j}(x)$. If $x \in X_T^j$, we do not need the random access for x in subsystem j.

3. Compute the grade $\mu_Q(x) = t(\mu_{A_1}(x), .., \mu_{A_m}(x))$ for each image x that has been "visited".

   Sort the images on their grades, and select the top k images for output.

3.1. Performance Analysis of Fagin's Algorithm

The cost of processing the query is measured in terms of the number of accesses to the database. There are two types of accesses: *sorted accesses* in phase 1 above and *random accesses* in phase 2. The *sorted access cost* is the total number of objects accessed from the database in phase 1. The *random access cost* is the total number of objects accessed from the database in phase 2. For example, if there are 100 images obtained from the color subsystem and 200 images from the texture subsystem in the sorted order, then the sorted access cost of the algorithm is 300. The *database access cost* is the sum of the random access cost and the sorted access costs.

Suppose a given query has m atomic conditions (query terms) and it is required to retrieve k top(best matching) images from the database. Let $S(i)$ and $R(i)$ denote the sorted and random access costs of subsystem i. The total database access cost $C(m,k,N)$ is given by,

$C(m,k,N) = \sum_{i=1}^{m} \{S(i) + R(i)\}$.

Images that have been accessed in phase 1 (sorted access) from one subsystem need not to be accessed again under random access from other subsystems. Thus the $R(i)$ is given by,

$R(i) = \sum_{i=1}^{m} \sum_{j=1}^{m} \|I_{S(i)} \cap I_{S(j)}\|$

where $I_{S(i)}$ is the set of images accessed in phase 1 (sorted access) in subsystem i.

Suppose we assume that the sorted list returned by subsystem i (say, based on color feature matching) is completely independent of the sorted list returned by subsystem j (say, based on texture feature). This means that the m atomic queries are independent of each other. Under this model, Fagin has given a probabilistic bound for the cost of processing the query.

Theorem 1 *The database access cost of single-step algorithm is $O(N^{(m-1)/m} k^{1/m})$ with arbitrarily high probability, where N is the number of images in the database.*

Proof: Refer [5]

The above bound is quite significant but discouraging (for large m cost is almost linear). But there are a number assumptions to be considered in detail, and the cost need be analyses further for its practical significance. First, the database images can be indexed, in CBIR systems such as CHITRA, on their OID using a hash table. Thus we can perform random access in constant time. The question, how realistic is the "independence" assumption need be considered.

The worst case cost being almost linear, is discouraging for implementers of query optimizer. To better understand the nature of the query processing problem, we carry out an analysis. We first consider the probability distribution, and the expected value of the cost of the single-step algorithm. For a given query Q with m terms, suppose it is required to retrieve top k images from a database of N images. Let $P(N,m,T,k)$ represent the probability that we need to make T accesses to the database in phase 1 (sorted access) of the algorithm. Such probabilities are called as committee problems in classical combinatorial probability theory. The above probability is same as the probability of choosing m committees from a population N, where each committee size is T and k is the number of people common to all (k are in all) committees. We can derive an expression for the probability as follows using the results from combinatorial probability theory [7].

The total number of ways in which m committees, each having T members can be chosen is $= \binom{N}{T}^m$.

We are interested in the number of committees, all of which have exactly k common members.

We assume the member's are numbered $1, 2, ...N$. Let S_i denote the set of choices of the committees, in which the person number i is common.

By inclusion-exclusion,

$S_i = \sum_{r=k}^{T} \binom{r}{k} (-1)^{(r-k)} A_r$,

where A_r is the sum over all sets R of r people of the number of choices of committees containing R in the intersection.

$A_r = \binom{N}{r} \binom{N-r}{T-r}^m$

Hence, the required probability is given by,

$P(N,m,T,k) = \frac{\sum_{r=k} T \binom{r}{k} (-1)^{(r-k)} \binom{N}{r} \binom{N-r}{T-r}^m}{\binom{N}{T}^m}$

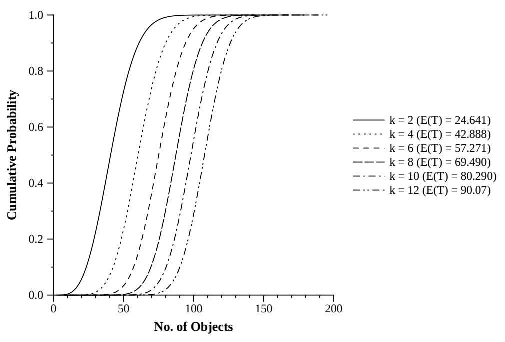

Figure 1. Probability distribution of T for m= 2, k =10 and N =1000

Color =	"red"	texture =	"fine"	Result	
objid	Grade	objid	Grade	objid	Grade
01	0.9	04	0.5	04	0.5
02	0.8	03	0.45	03	0.45
03	0.7	05	0.4	02	0.3
04	0.5	02	0.3	01	0.2
05	0.1	01	0.2	05	0.1

Figure 2. Sample data for an example query, "color = ***red*** **AND texture =** ***fine*****"**

When we have made T accesses(in phase 1) to the data base, the probability $P_s(T)$ of having retrieved at least k common images is given by the summation,

$P_s(T) = (\sum_{j=k}^{j=T}(P(N,m,T,j)),$

The probability $P_u(T)$ of failing to retrieve k common images after $T-1$ accesses is given by the product,

$P_u(T) = P_u(T-1) * (1 - P_s(T)).$

The expected value of T, $E(T)$ is then,

$E(T) = (\sum_{T=k}^{T=N} t * P_s(T) * P_u(T-1).$

Note that $P_u(k-1) = 1$ in the above expression.

Figure 1 shows the probability distribution of T (number of images accessed in phase 1 of the algorithm) for $N = 1000$, $m = 2$ and $k = 2, 4, 6, 8, 10$ and 12. We observe that the distribution appears to be a double exponential having a narrow spread. The cost of processing queries tends to be with in a small range rather than a distribution. The expected/worst/best cost tend to be close to each other. These observations are based on the results obtained with $m = 2$. Further work is required in this area with higher values of m.

4. Multi-Step Query Processing Algorithm

We were motivated by the observation that Fagin's algorithm requires more database accesses than necessary, when the lower and upper bound properties are satisfied by the combining function used. Consider the example shown in Figure 2, where N = 5 and m = 2. Suppose the user wants to retrieve best two matching images ($k = 2$) the single-step algorithm requires 4 accesses from each subsystem($T = 4$). The total database access cost is then 10 (8 sorted accesses and 2 random accesses). We observe that, we only need 2 accesses from each list, if the combining function used satisfies the bounding properties. The corresponding database access cost is 8 (4 sorted + 4 random).

As already mentioned, in CBIR systems like CHITRA the random access cost is much smaller (a constant if hash tables are used) as compared to the sorted access cost. Thus the number of sorted accesses is the major cost factor of single-step query processing algorithm. It appears difficult to formalize an absolute optimality criteria. Clearly *the optimal* algorithm will make nonuniform accesses to the subsystems. Such algorithms are hard to find and are likely to be based on heuristics. However, we define an optimality criterion for a class of algorithms that accesses elements uniformly in all subsystems (makes equal number of requests to all subsystems for sorted list). We then present a multi-step algorithm that requires smaller number of sorted accesses when the combining functions used satisfy the bounding properties. We show that our multi-step algorithm satisfies the optimality criterion. In this paper, we discuss the algorithms for conjunctive queries. The same is valid for disjunctive queries as well.

4.1. Optimality Criterion

Our following definition of T-Optimality criterion is based on the premise that the number of sorted accesses should be minimum.

Definition 4 T-optimality: *A query processing algorithm is T-optimal if T is the minimum number of sorted accesses required for any uniform algorithm (to retrieve top k elements, accessing T elements from each of the subsystems).*

Lemma 1 *For an algorithm that uses a combining function t which satisfies the lower bounding property, let t_h be the threshold value at iteration T. Then, t_h can be evaluated as, $t_h = t(\mu_{A_1}(x_1^T), .., \mu_{A_m}(x_m^T))$. Then for any image x that has not been retrieved so far, $\mu_Q(x) \leq t_h$.*

Proof: Let y be an element that has not been retrieved after T^th iteration. Assume that $\mu_Q(y) > t_h$. Since each subsystem outputs elements in sorted order, $\exists i : \mu_{A_i}(y) \leq t_h$. By the definition of lower bounding property it follows that, $\mu_Q(y) \leq t_h$. This contradicts with our assumption, and hence $\mu_Q(y) \leq t_h$.

Lemma 2 *An algorithm is $T - optimal$ if and only if after accessing T elements from each subsystem, for all images y that have been retrieved $|\mu_Q(y) > t_h^T| \geq k$ and $|\mu_Q(y) > t_h^{T-1}| < k$.*

Proof: This follows from definition 4 and lemma 1.

As shown by the preceding example, the single-step query processing algorithm is not $T - optimal$ as it requires more accesses to the subsystems. In other words, the need of k common elements in phase 1 of the algorithm leads to the violation of $T - optimality$ criteria. Our multi-step algorithm avoids this, and is $T - optimal$.

4.2. The Multi-step Algorithm and it's Analysis

The following multi-step query processing algorithm returns the top k images for a CBIR query $Q = F_t(A_1, .., A_m)$. As explained above, each subsystem i, $1 \leq i \leq m$, returns the matching images in sorted order.

1. For each query term A_i, request the subsystem i to return the next image x. Thus, each subsystem i outputs the graded set of pairs $(x, \mu_{A_i}(x))$, where x is an image in the database and $\mu_{A_i}(x)$ is the similarity value of x under A_i.
2. For each image x returned by the subsystem i in the current iteration, do random access to subsystems j, $i \neq j$, to find $\mu_{A_j}(x)$.
3. Compute the threshold grade, t_h for this iteration, $t_h = t(\mu_{A_i}(x_i), .., \mu_{A_m}(x_m))$. Here x_i is the element retrieved by subsystem i in the current iteration.
4. Compute the grade $\mu_Q(x) = t(\mu_{A_i}(x), .., \mu_{A_m}(x))$ for each image x retrieved in this iteration. Update Y, the set containing all images that have grade $\mu_Q(x) \geq t_h$.
5. Go to next iteration (repeat steps 2 to 5) until the set Y has k images.
6. Output the graded set $\{(x, \mu_Q(x) | x \in Y\}$.

We now prove the correctness of our algorithm.

Theorem 2 *For every query $F_t(A_1, .., A_m)$, the multi-step algorithm correctly returns the top k answers, if t satisfies the bounding properties.*

Proof: It is clear that Y has k members. Assume that there exists an image z which should have been in Y, but is not. That is, for some y in Y, $\mu_Q(y) < \mu_Q(z)$.

Case I: Since the combining algorithm satisfies the lower bounding property, $\mu_Q(y) < \mu_Q(z)$ implies that y has at least one μ_{A_i} that is smaller than all μ_{A_i} of z. By definition of threshold, $\mu_Q(z) \leq t_h$, and $\mu_Q(y) \geq t_h$. That contradicts, our assumption $\mu_Q(y) < \mu_Q(z)$ based on Lemma 1.

Case II: Since the combining algorithm satisfies the upper bounding property, $\mu_Q(y) < \mu_Q(z)$ implies that z has at least one μ_{A_i} that is greater than all μ_{A_i} of y. By definition of threshold, $\mu_Q(z) \leq t_h$, and $\mu_Q(y) \geq t_h$. That contradicts, our assumption $\mu_Q(y) < \mu_Q(z)$.

Theorem 3 *The multi-step query processing algorithm is $T - optimal$*

Proof: Follows from the definition of bounding properties and Lemma 1 and 2.

Proposition 1 *If the number of common elements retrieved in sorted accesses by the multi-step algorithm is k', and that by single-step algorithm is k, then $k' \leq k$ always.*

Proof: This follows from theorem 3 and lemma 2.

Theorem 4 *Under the probabilistic model, the database access cost of the multi-step algorithm $C(m, k, N)$ is bounded by $O(N^{(m-1)/m} k'^{1/m})$ with the arbitrarily high probability. Here k' is the number of common elements retrieved by the subsystems under sorted access.*

Proof: follows from proof in [5] and proposition 1.

Proposition 2 *The database access cost for multi-step query processing algorithm is always less than or equal to single step algorithm, that is, $O(N^{(m-1)/m} k'^{1/m} \leq O(N^{(m-1)/m} k^{1/m})$.*

Proof: From the above proposition, $k' \leq k$. Thus, $O(N^{(m-1)/m} k'^{1/m} \leq O(N^{(m-1)/m} k^{1/m})$.

5. Experimental Results

In order to study the relative performance of the algorithms in practice, we have implemented them on our CHITRA system. The algorithms were implemented in C, and the experiments were run on an Sun Solaris Unix machines. We used a set of 1000 images obtained from the World Wide Web for our test data. Each image is of size 192×128 and in .ppm format. We ran two sets of experiments using different features and similarity functions. For the cost involved, we only consider the number of accesses to the subsystem, as analysed (and not the exact number of disk accesses by the subsystem).

5.1. Color and Texture Features

For our first set of experiments, we extracted and used color and texture features. The color feature of each image is represented by a histogram of size 768(256 for each

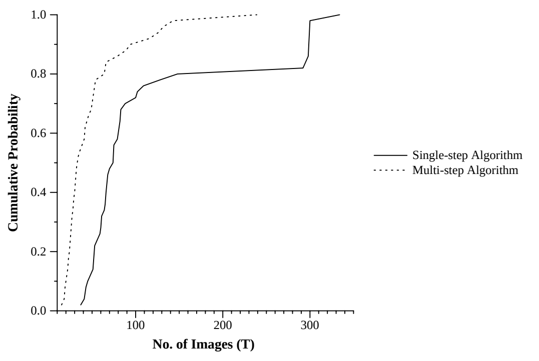

Figure 3. Probability distribution of T for m= 2, k =10 and N =1000

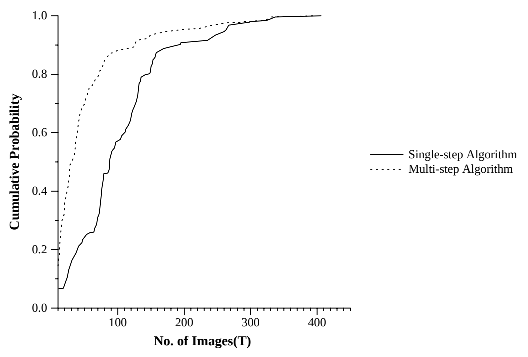

Figure 5. Probability distribution of T for $m = 2$, $k = 10$ and $N = 1000$

R,G,B) elements. Gabor functions were used to obtain texture feature, each image represented by a 48 dimensional vector [8]. The query vectors were obtained by randomly selecting an image from the database and extracting the corresponding features. Thus the query posed is "retrieve all images similar to the given image (by using the corresponding color vector and texture vector)". We then noted the number of sorted accesses, random accesses and total accesses required to the subsystems to retrieve the $k = 10$ best matches. The experiment was repeated 50 times, for 50 different query images selected randomly from the database. Using these results, Figure 3 plots the cumulative probability distribution of the required number of sorted accesses (T) to the subsystems (for $N = 1000$, $m = 2$, and $k = 10$) for each of the two algorithms. We repeated the experiments for various values of k, and consistent results were obtained as shown in Figure 4 which lists the various access costs for the two algorithms. We note that the multi-step algorithm clearly outperforms the single-step algorithm by about 50%.

5.2. Color Feature Based on Human Perception

The purpose of this set of experiments was to evaluate our algorithms for larger number of subsystems, $m > 2$. Towards this end, we used a different set of features for the same set of images. Carson and Ogle experimented in Berkeley digital library project, and concluded that only 13 colors were "distinguishable" by humans, and devised a system of color features using these 13 colors [2, 10]. We extracted these color features from our images and used for our experiments. With these features, the user can pose queries such as $< red, mostly >$ and $< green, few >$ to retrieve images which are mostly red, and a little green. We posed 500 such different queries, each having up to $m = 5$ query terms. The queries were processed using both single-step and multi-step algorithms and results were noted as before. Figure 5 plots the cumulative probability distribution of the number of sorted accesses to the subsystem required to retrieve the top $k = 10$ images. Figure 6 shows the access costs for various values of m, and $k = 10$. We observe that multi-step algorithm consistently outperforms the single-step algorithm. Note that for larger values of m, the cost of the algorithms exceeds N, and is an inherent limitation of the processing algorithms. Obviously, in practice one would rather use sequential scanning in such cases. Our feeling is that there are heuristics to be used, which we are investigating further, to achieve much better performance in practice.

6. Conclusions

In this paper we addressed a query processing problem encountered in CBIR systems, such as our CHITRA system under development. The problem is to retrieve best matching images corresponding to the complex query posed by the user. There has been considerable research in access structures, to process simple queries (single feature). Only recently researchers are addressing the problem of processing complex queries.

We first defined the frame work of our problem. A probabilistic analysis of Fagin's single-step algorithm supplementing his bounds were given. With the insight gained, we arrived at a new multi-step algorithm. We derived some bounds for the new algorithm. Experimental results, on our CHITRA test bed with real life data, were presented for two different feature systems showing the multi-step algorithm performs 50% better than the single step algorithm. Clearly there is much scope for the use of heuristics to over come the inherent complexity of the problem and we are investigating the heuristics further.

Acknowledgments: Help provided by Lioudmila Volo-

k	Single-step Algorithm				Multi-step Algorithm			
	expected	sorted	random	total	expected	sorted	random	total
2	59.61	119.23	23.58	142.81	28.17	56.36	21.48	77.84
4	83.37	166.75	62.04	228.79	38.15	76.32	31.48	107.80
6	96.76	193.52	83.26	276.78	40.41	80.83	34.58	115.41
8	107.19	214.39	99.32	313.71	45.82	91.63	44.25	135.88
10	116.68	233.36	113.59	346.95	49.71	99.44	50.79	150.23
12	125.39	250.8	125.19	375.99	51.15	102.32	53.16	155.48

Figure 4. Number of database accesses for the two algorithms, $m = 2$.

m	Single-step Algorithm				Multi-step Algorithm			
	expected	sorted	random	total	expected	sorted	random	total
2	104.165	208.33	187.3	395.63	52.96	105.92	100.49	206.51
3	189.85	569.56	725.148	1294.70	101.22	303.68	460.78	764.46
4	249.59	998.35	1399.17	2397.52	152.45	609.79	1078.69	1688.48
5	288.35	1441.77	2058.63	3500.40	191.57	957.88	1750.65	2708.53

Figure 6. Number of database accesses for the two algorithms, $k = 10$. Please see the explanation about cost exceeding $N = 1000$.

dina of RMIT and Nick Wormald of The University of Melbourne in solving the probability problem in section 3 is gratefully acknowledged.

References

[1] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 322–331, Atlantic City, NJ, 1990.

[2] C. Carson and V. E. Ogle. Storage and retrieval of feature data for a very large online image collection. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 19(4):19–27, December 1996.

[3] S. Chaudhuri and L. Gravano. Optimizing queries over multimedia repositories. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, pages 45–52, 1996.

[4] W. F. Cody, L. M. Haas, W. Niblack, M. Arya, M. J. Carey, R. Fagin, M. Flickner, D. Lee, D. Petkovic, P. M. Schwarz, J. Thomas, M. T. Roth, J. H. Williams, and E. L. Wimmers. Querying multimedia data from multiple repositories by content: the Garlic project. Third Working Conference on Visual Database Systems (VDB-3), Lausanne, Switzerland, March 1995.

[5] R. Fagin. Combining fuzzy information from multiple systems. Proc. Fifteenth ACM Symp. on Principles of Database Systems, pages 216–226, Montreal, 1996.

[6] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *Computer*, 28(9):23–32, September 1995.

[7] C. Liu. *Introduction to Combinatorial Mathematics*. McGraw-Hill Book Company, 1968.

[8] B. Manjunath and W.Y.Ma. Texture features for browsing and retrieval of image data. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 18(8):837–842, August 1996.

[9] S. Nepal, M.V.Ramakrishna, and J.A.Thom. Four layer schema for image data modelling. In C. McDonald, editor, *Australian Computer Science Communications, Vol 20, No 2, Proceedings of the 9th Australasian Database Conference, ADC'98*, pages 189–200, 2-3 February, Perth, Australia, 1998.

[10] S. Nepal, M.V.Ramakrishna, and J.A.Thom. A fuzzy system for content based image retrieval. In *Proceedings of the Second IEEE International Conference on Intelligent Processing Systems*, pages 335–339, 4-7 August, Gold Coast, Australia, 1998.

[11] G. Salton. *Automatic Text Processing: the transformation, analysis, and retriev al of informaiton by computer*. Addison-Wesley, Reading, MA, 1989.

[12] S. Santani and R. Jain. Similarity queries in image databases. In *Proceedings of CVPR96, IEEE International Conference on Computer Vision and Pattern Recognition*, San Francisco, June 1996.

[13] J. R. Smith and S.-F. Chang. VisualSEEk: A fully automated content-based image query system. In *ACM Multimedia*, Bonton, MA, November 1996.