# APPROXIMATING THE SELECTIVITY OF MULTIMEDIA RANGE QUERIES

*Mario Döller*

Department of Information Technology
University Klagenfurt, Austria
mdoeller@itec.uni-klu.ac.at

*Harald Kosch*

Department of Information Technology
University Klagenfurt, Austria
harald@itec.uni-klu.ac.at

## ABSTRACT

*This paper introduces a new approach of approximating the selectivity of multimedia range queries. Estimating the selectivity of a range query is a pre-requisite to optimize a multimedia database query. We use the DBSCAN clustering technique for finding high density areas in the data set. Then, the selectivity is approximated with the help of a density function in combination with the volume of the query's hypersphere. Our approach is fast and accurate which was evaluated on an image data set using the MPEG-7 Scalable Color Descriptor. The technique is integrated with the help of the extensible optimizer architecture in the Oracle multimedia database system.*

## 1. INTRODUCTION

Nearest-Neighbor and Range queries are very common in multimedia databases. The performance of such queries can be increased by the use of appropriate index structures and by the use of an query optimizer that takes care of the optimal query plan. A cost-based query optimizer uses two factors for finding an optimal query plan: on the one hand, there is the *cost* value for an operation which is composed of the amount of CPU cycles that is necessary to perform the operation, the used network bandwidth and the number of necessary page accesses; on the other hand, the *selectivity* of an operation which is defined as the ratio of output relation versus input relation of the operation. The estimation of the costs for the NN- and Range Searches are well investigated [2, 3]. The estimation of a Range query's selectivity is, apart from few initial approaches [1], an open research question.

In this context, we introduce an approach of approximating the selectivity of Range Searches within a n-dimensional feature space. This paper is organized as follows: Related work about query optimization and clustering is handled in Section 2. Our approach of approximating the selectivity is formulated in Section 3. The evaluation of our approach is discussed in Section 4.

## 2. RELATED WORK

In the literature, several cost models exist that concentrate on calculating the amount of page access for range- and k-NN-queries (see [2, 3]). Selectivity estimation [1] for range queries is often limited to either the assumption of uniformity and independence of data sets or to the low-dimensional spaces, e.g., 2- or 3- dimensional data spaces in geographic information systems. In any case, the uniformity assumption does not hold for real data sets and is thus not relevant for us.

In [8], the authors present an efficient cost model for predicting the performance of the k-NN-query independently of the used index tree. The model is accurate for low- and mid-dimensional data with non-uniform distributions. For this purpose, the authors introduce two new concepts: the regional average volume and the density function.

As our approach bases on clustering techniques related work in this area is provided as well. In recent years, a number of clustering algorithms have been introduced [6, 5]. In addition, there exist several excellent surveys on clustering [7]. But yet, clustering has not been applied to multimedia query optimization.

## 3. APPROACH

As stated before, a fast prediction of the selectivity for a range query highly depends on the underlying data set. The selectivity is related to the notion of density in the neighborhood of the query point, for instance, if the density around the query point is high, we have a high selectivity, otherwise a low density will lead to a small selectivity.

In this context, we originally introduce an approach of approximating the selectivity of Range-Searches within a n-dimensional data set with the help of a density based clustering technique (in our case DBSCAN). The DBSCAN algorithm was used because it is a density based and wide spread clustering technique. In addition, the *eps*-value of the algorithm is an important component of our approach and allows us to optimize the cluster allocation for each query circle radius given. Two approaches were implemented. In the first

*0-7803-9332-5/05/$20.00 ©2005 IEEE*

approach we applied the clustering algorithms to the dataset without taken into consideration possible query circle radii. This approach worked well for certain queries only: the approximated selectivity showed only an acceptable failure for a small range of query circle radii. Thus, we come up with a second and advanced approach that removes the main problems of the first one. In example, the best *eps*-value is determined for a discrete query circle radius depending on a given query circle radius interval.

### 3.1. First Approach

The key idea of our first approach includes the following steps:

1.) cluster the data set with the help of a density based cluster technique (in our case DBSCAN [5]).

2.) specify the MBR (minimum bounding region) for every cluster C

3.) calculate the density of all clusters with:

$$Density(C) = \frac{\text{\#P of C}}{Vol(C)} \tag{1}$$

where Vol(C) is the hypervolume of the surrounding MBR and #P denotes the number of points that are in the cluster. Note: We assume uniform distribution within a cluster.

4.) approximate the selectivity of a query point Q with radius r with the following formula:

$$approx.selectivity = \begin{cases} Density(C_i) * Vol(Q) & \text{, Q} \in Area(C_i) \\ MinPts - 1 & \text{, otherwise} \end{cases} \tag{2}$$

where Vol(Q) is the hypervolume of the surrounding sphere defined through the query point Q and radius r. The approximated selectivity is determined by $Density(C) * Vol(Q)$ if and only if the query point Q falls in the area of cluster C. The area of the respective cluster is specified by the MBR. If the query point Q falls in none of the given clusters, then the approximated selectivity is defined by *MinPts - 1*. The *MinPts* value is equivalent to the DBSCAN *MinPts* value and specifies the absolute minimum amount of points a cluster has to have. Therefore, we can argue that, if the query point does not fall in a predefined cluster then the density around the query point is rather low.

We used several data sets for the evaluation of our approach (see [4] for a detailed explanation). The experiments showed that for a query circle radius which is near to the used *eps*-value for clustering the data set, the approximated result has only a failure around 12% of the average real result. Unfortunately, the failure between real and approximated selectivity for other query circle ranges is about 50% which is unacceptable. The failure was calculated with the following formula:

$$failure = \frac{\sum_{i=1}^{n} |RealSel(Q_i) - ApproxSel(Q_i)|}{n} \tag{3}$$

where $Q_i$ is the i-th point of the data set used as query point. *RealSel(Q)* is the result of the real selectivity and *ApproxSel(Q)* is the approximated result of our approach.

### 3.2. Second Approach

The main weakness of the former approach was the usage of the same *eps*-value and therefore the same cluster allocation for each query circle radius. This results in a good performance for a small range of query circle radii and a worse performance for the others. Hence, our second approach identifies and assigns the failure-minimal cluster allocation to the corresponding query circle radius which leads to an acceptable performance for the whole range of given query circle radii. The major changes in contrast to the previous approach concern the first step. In the following, all steps of the previous approach are traversed and the necessary changes are explained.

1.) At first, the algorithm identifies the failure-minimal cluster allocation for all discrete query circle radii depending on the given interval and the appropriate incremental factor. The input (given by the user) for the algorithm is the minimum query circle radius (*min_qcr*), the maximum query circle radius (*max_qcr*) and the incremental factor *incr_factor*. This information relies on the underlying data interval and is used to decrease the size of the search space.

Step 1: Here, all discrete query circle radii of the given query circle radius range are determined. For instance, for the following given data: *min_qcr*=0.04, *max_qcr*=0.10 and *incr_factor*=0.01, we result in 7 intermediate radii (0.04, 0.05, ... , 0.10 ).

Step 2: Starting from the given *min_qcr, max_qcr* and *incr_factor* values, we evaluate the necessary *min_eps* and *max_eps* values for computing the different cluster allocations. The *min_eps* value is composed of *min_qcr - to_add*, respectively the *max_eps* value is determined by *max_qcr + to_add*. The *to_add* value is calculated with the following formula:

$$to\_add = \frac{\text{max\_qcr} + \text{min\_qcr}}{2} * 0.7 \qquad (4)$$

The *min_eps*-value can not be negative and the *max_eps*-value has its upper boundary when all points are in one cluster. In addition, as in step 1, all necessary intermediate *eps*-values are determined.

Step 3: In the last step, the best cluster allocation for a certain discrete query circle radius is determined. This is done by evaluating the failure of all cluster allocations (see Equation 3).

For this purpose, the second and third task, namely specify the MBR (task 2) and the density (task 3) for each cluster of the previous approach are integrated in this step. Then, the failure is calculated by summing up the differences between the approximated selectivity and the real selectivity for all points within the data set. The cluster allocation with the minimum failure is assigned to the corresponding query circle radius.

4.) As already mentioned, the second and third task of the previous approach are already integrated without changes in the step above. For approximating the selectivity, the original fourth task has to be modified slightly. Now, the algorithm uses the given query circle radius for identifying the appropriate cluster allocation. Then, the approximated selectivity is calculated with formula 2 (see fourth point of the previous approach) with the given query point and radius and the assigned cluster allocation.

## 4. EVALUATION

This section describes the series of experiments we performed in order to evaluate the effectiveness of our approach. The tests were carried out on MPEG-7 Scalable Color Descriptors (of dimension 8) extracted from an image data set. Scalable Color Descriptors are principally color histograms which have been treated by a Haar transformation before. Additional tests can be found at [4]. Our evaluation pursued two targets: First, we compared our approximated selectivity to the real selectivity. Second, we show the performance gain of our approach within the Oracle database. The experimental settings are as follows:

- The 8-dimensional color feature vectors are extracted according to the definition of the MPEG-7 Scalable Color Descriptor from images of size 128x192 pixels. The values are standardized to be within the [0..1] interval.

### 4.1. Detailed Results

In order to verify our approach, we calculated the failure (see Equation 3) between the real selectivity and the approximated selectivity over all points (2000) in the data set.

Figure 1 shows the maximum real selectivity (boxed line), the average real selectivity (triangle line) and the failure between real selectivity and approximated selectivity (diamond line). Figure 2 shows the average real selectivity (boxed line) and compares it to the average approximated selectivity (diamond line). The query circle radius of the image test-bed ranges between 0.04 to 0.15 with an incremental factor of 0.01.

The evaluation of our test-bed results in: The average relative failure which is calculated by the sum of all percentages of average real selectivity compared to their corresponding failure yields an average percentage of 11% (0.07% - 30%).

This means that on average, for all query circle radii, the difference of the approximated selectivity to the real selectivity is only around 11% which is an acceptable result for an approximation. This result is confirmed by the analysis of Figure 2 which results in an average difference of 4% for the images test-bed. This means that the difference between the average real selectivity and the average approximated selectivity over all points of the data set is small.

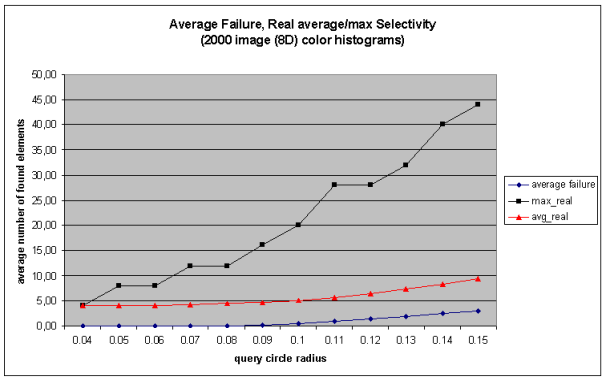


**Fig. 1**. Average Failure Evaluation

#### 4.1.1. Database evaluation

We used the following setting for the evaluation of our approach within the database. The test-bed contains two tables and simulates a geographic application: The first table, named *city*, contains an unique ID and the name of an city. The second table, named *location*, contains the foreign key ID of the table *city* and the location of the city given as 2-dimensional coordinates. Both tables contain 1000 tuples.

The analysis was twofold. First, the changes of the query plan were evaluated with the help of the *EXPLAIN PLAN*

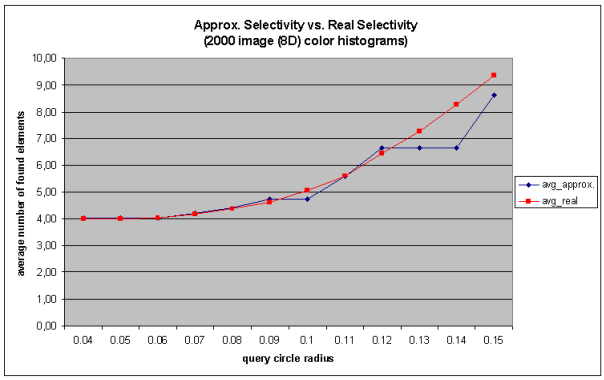


**Fig. 2**. Approximated Selectivity Evaluation

| without an approximation | with an approximation |
|---|---|
| QUERY PLAN | QUERY PLAN |
| SELECT STATEMENT | SELECT STATEMENT |
| SORT AGGREGATE | SORT AGGREGATE |
| NESTED LOOPS | NESTED LOOPS |
| TABLE ACCESS FULL CITY | TABLE ACCESS FULL LOCATION |
| TABLE ACCESS FULL LOCATION | TABLE ACCESS FULL CITY |

**Table 1**. Query Plan Comparison of second query

command. Second, the execution times for the queries in combination with and without our approximation are compared.

In the following, we are interested in all cities that are in a specific area or whose name starts with an A. This would result in the following query:

```
explain plan SET STATEMENT_ID = 'selectivity'
for
select count(*) from location l, city c
where (rangeQueryOp(l.location, '50.0 10.0', 2, 0.22) = 1
       OR c.name LIKE 'A%')
       AND l.cityid = c.id;
```

The acquired query plan is presented in Table 1. The left column denotes the query plan without an a priori information of the selectivity and the right column shows an improved query plan. The crucial factor for the performance is the ordering of the inputs to the join operator. Without a priori information, the city table is left to the join. In fact, the amount of resulting tuples of the selection operation for the city table is much higher than for the range query operation on the location table. Therefore, the right column of the table shows the best performance. This is also stated by the real measurement results which is 89.2 seconds for the query plan of the left column and 45.8 seconds for the plan of the right column.

## 5. CONCLUSION

This paper introduced our approach for approximating the selectivity of range queries. We used the DBSCAN clustering technique for finding high density areas in the data set. The selectivity was approximated with the help of a density function in combination with the volume of the query's hypersphere.
The approach was evaluated on MPEG-7 Scalable Color Descriptors extracted from an image data set. The evaluation showed that on average the difference to the real selectivity was minimal and did not exceed 1/3. In nearly all cases, the relative failure is around 11%. Finally, it was shown that the query plan and hence the performance of queries was improved by the use of our approach. Our approach is integrated into the Oracle Multimedia Database System and works together with other system components of the database system.

## 6. REFERENCES

[1] Swarup Acharya, Viswanath Poosala, and Sridhar Ramaswamy. Selectivity Estimation in Spatial Databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*, pages 13–24. ACM Press, 1999.

[2] Stefan Berchtold, Christian Böhm, Danial A. Keim, and Hans-Peter Kriegel. A Cost Model for Nearest Neighbor Search in High-Dimensional Data Space. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 78–86, Tucson, Arizona, United States, 1997.

[3] Christian Böhm. A Cost Model for Query Processing in High Dimensional Data Spaces. *ACM Transactions on Database Systems (TODS)*, 25(2):129–178, 2000.

[4] Mario Döller. *The MPEG-7 Multimedia DataBase System (MPEG-7 MMDB)*. Dissertation, University Klagenfurt, Austria, 2004.

[5] Martin Ester, Hans-Peter Kriegel, Jrg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, OR, USA, 1996.

[6] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. CURE: An Efficient Clustering Algorithm for Large Databases. In *Proceedings of the ACM SIGMOD Conference*, pages 73–84, 1998.

[7] S. B. Kotsiantis and P. E. Pintelas. Recent Advances in Clustering: A Brief Survey. *WSEAS Transactions on Information Science and Applications*, 1(1):73–81, 2004.

[8] Ju-Hong Lee, Guang-Ho Cha, and Chin-Wan Chung. A Model for k-Nearest Neighbor Query Processing Cost in Multidimensional Data Spaces. *Information Processing Letters*, 69(2):69–76, 1999.