Efficient Processing of Complex Similarity Queries in RDBMS through Query Rewriting

Caetano Traina Jr.
Dept. of Computer Science
ICMC-Univ. of Sao Paulo
Sao Carlos, SP, 13560-Brazil
caetano@icmc.usp.br

Agma J. M. Traina
Dept. of Computer Science
ICMC-Univ. of Sao Paulo
Sao Carlos, SP, 13560-Brazil
agma@icmc.usp.br

Marcos R. Vieira
Dept. of Computer Science.
ICMC-Univ. of Sao Paulo
Sao Carlos, SP, 13560-Brazil
mrvieira@icmc.usp.br

Adriano S. Arantes
Silicon Valley Laboratory
IBM
San Jose, CA
aarante@us.ibm.com

Christos Faloutsos
Dept. of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
christos@cs.cmu.edu

ABSTRACT

Multimedia and complex data are usually queried by similarity predicates. Whereas there are many works dealing with algorithms to answer basic similarity predicates, there are not generic algorithms able to efficiently handle similarity complex queries combining several basic similarity predicates. In this work we propose a simple and effective set of algorithms that can be combined to answer complex similarity queries, and a set of algebraic rules useful to rewrite similarity query expressions into an adequate format for those algorithms. Those rules and algorithms allow relational database management systems to turn complex queries into efficient query execution plans. We present experiments that highlight interesting scenarios. They show that the proposed algorithms are orders of magnitude faster than the traditional similarity algorithms. Moreover, they are linearly scalable considering the database size.

Categories and Subject Descriptors: H.2.4 Database Management, Systems: Query processing

General Terms: Algorithms, Theory.

Keywords: Similarity predicates, query rewriting.

1. INTRODUCTION

Exact match comparisons are rare in domains of complex data, such as images, videos, genomic sequences, and time series. Comparison based on ordering relationships cannot be applied either, as the total ordering property does not hold. Therefore, similarity emerges naturally as the natural way to compare data in complex domains.

Similarity comparisons employs Distance Function (DF) to quantify how similar, or close two objects are. Those functions are the basis to create a metric space, which formally is a pair $\mathbb{M} = <\mathbb{S}, d()>$, where $\mathbb{S}$ denotes the universe of valid elements and $d()$ is the function $d\colon \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$ that expresses a "distance" between elements of $\mathbb{S}$, i.e., the smaller the distance, the closer or more similar the elements are [9]. A DF must satisfy the properties of: **symmetry**: $d(s_1, s_2) = d(s_2, s_1)$; **non-negativity**: $0 < d(s_1, s_2) < \infty$ if $s_1 \neq s_2$ and $d(s_1, s_1) = 0$; and **triangular inequality**: $d(s_1, s_3) \leq d(s_1, s_2) + d(s_2, s_3)$, where $s_1, s_2, s_3 \in \mathbb{S}$. A data set S is said to be in a metric space if $S \subseteq \mathbb{S}$.

There are two basic similarity predicates, expressed as $\widetilde{\sigma}$: the range and the $k-$nearest neighbor queries. Given a data set $S \subset \mathbb{S}$ and a query center $s_q \in \mathbb{S}$, they are defined as:

1. **Range Query -** *Rq*: given the maximum search distance r_q, the range query $\widetilde{\sigma}_{\left(Rq(s_q, r_q)\right)} S$, expressed by the $Rq(s_q, r_q)$ predicate, selects every element $s_i \in S$ such that $d(s_i, s_q) \leq r_q$;
2. **k-Nearest Neighbor Query -** *kNN*: given an integer value $k \geq 1$, the k-nearest neighbor query $\widetilde{\sigma}_{\left(kNN(s_q)\right)} S$, expressed by the $kNN(s_q)$ predicate, selects the k elements $s_i \in S$ whose distances to s_q are the smallest.

Most algorithms for similarity search consider each query as an isolated operation instead of predicates integrating more complex query expressions. Moreover, they do not take advantage of optimizations that can be performed combining simpler expressions.Therefore, queries composed of multiple similarity predicates require expensive union and intersection operations to combine intermediate results to answer conjunctions and disjunctions of elementary predicates.

A RDBMS supporting multimedia data demands efficient ways to answer complex similarity queries. To integrate similarity queries with the traditional ones, the similarity predicates should be included into the relational algebra. As a consequence, similarity operators should be used together with other predicates already available in the relational algebra, such as exact match and order-based comparisons for textual/numeric attributes, and also in comparisons combining two or more similarity predicates. In this paper we delve into the later kind of queries, that is, those expressed as sequences of conjunctions and disjunctions of the basic similarity predicates. Examples of real systems demanding such kind of queries follows.

Q1: In a user friendly word processor: "*When a wrong word is written, show up to 5 correct words that differ at most 2 characters*" (k-NN and Range):
$\widetilde{\sigma}_{\left(5NN(<wrong\,word>)\right)}S \cap \widetilde{\sigma}_{\left(Rq(<wrong\,word>,2)\right)}S$.

Q2: In real state business: *"Show the 10 nearest available houses to my job 'mj' that are not farther than 15 miles from 'mj', and not farther than 15 miles from my wife's job 'wj'* (conjunction of several predicates):
$(\widetilde{\sigma}_{\left(10NN(mj)\right)}S \cap \widetilde{\sigma}_{\left(Rq(mj,15mi)\right)}S) \cap \widetilde{\sigma}_{\left(Rq(wj,15mi)\right)}S$.

Q3: In health-care information systems: *"Show the XRay exams of any patient that are the 10 most similar to each of these three XRay exams e_1, e_2 and e_3 from my current patient but that do not differ from them more than 10%*" (disjunction of conjunctions):
$(\widetilde{\sigma}_{\left(Rq(e_1,10\%)\right)}S \cap \widetilde{\sigma}_{\left(10NN(e_1)\right)}S) \cup (\widetilde{\sigma}_{\left(Rq(e_2,10\%)\right)}S \cap \widetilde{\sigma}_{\left(10NN(e_2)\right)}S) \cup (\widetilde{\sigma}_{\left(Rq(e_3,10\%)\right)}S \cap \widetilde{\sigma}_{\left(10NN(e_3)\right)}S)$.

Another important application for complex similarity queries is to support relevance feedback (RF), which allows the users to set the elements of a query that fulfills his/her interest, guiding automatic corrections of future queries [16]. The query reprocessing can change the distance function or the center element. It can also take advantage of rewriting the similarity query including new similarity conditions. The later approach produces the best results, but rely on the ability to efficiently handle complex queries.

Although complex queries can be executed combining intermediate results of basic range and k-NN algorithms by set-theoretical operators, a more efficient approach should use few general algorithms configured by the query optimizer of a RDBMS. This approach leads to two core problems: how to rewrite a query plan to obtain an optimal execution; and how to make frequently-used composite operations more efficiently executed by a multi-purpose algorithm than by the sequential execution of the basic algorithms combined by set-theoretical operators. We tackle both problems in this paper, addressing the following issues:

1. Which multi-purpose algorithms are efficient to answer queries composed of conjunctions and disjunctions of similarity predicates?
2. What rules guide query rewriting when generating strategies to execute the similarity search operators?
3. How to adequately represent a query to be submitted to multi-purpose similarity search algorithms?

The remainder of this paper is structured as follows. The next section gives a brief history of algorithms for answering similarity queries. Section 3 describes the basic algorithms for similarity queries and their most common variations. Section 4 presents the rules governing complex similarity queries centered at the same query element and defines the algorithms required to support them. Section 5 presents rules governing complex similarity queries centered at different query elements and an algorithm to answer them. Section 6 show experimental results comparing the traditional approach with the algorithms and concepts presented in this paper. Finally, section 7 concludes this paper.

2. RELATED WORK

The properties of a distance function led to the development of hierarchical index structures called Metric Access Methods (MAM). MAM are fundamental to accelerate searches in large sets of complex data types, where only the set of elements and a DF are available. MAM such as the M-tree [9] and the Slim-tree [22] can accelerate similarity queries on complex data by orders of magnitude.

2.1 Similarity Search Algorithms

Similarity search algorithms motivated several works. Algorithms for range queries are straightforward, because a limiting radius is always known throughout its execution. However, the minimum radius that cover the k nearest elements is not known beforehand for k-NN algorithms. Thus, the ordering to search subtrees in hierarchical index structures is important, turning the choice of path sequences that lead to high subtree-pruning one of the most pursued objectives [22]. Many approaches have been proposed to improve the performance of k-NN queries as, for instance, branch-and-bound [17], incremental [11], multi-step [13, 18] and fast parallel algorithms [1]. Another approach is to estimate a final limiting range for the query and perform a sequence of "start small and grow" steps [20]. All of these works refer to algorithms dealing with just one simple similarity predicate.

Multiple similarity queries executed as a single command have been introduced, allowing potential for much more optimization than single queries do [4, 2]. Algorithms to answer multiple similarity queries substantially speed-up query-intensive data mining applications. Multiple k-NN queries are addressed in [4], analyzing optimizations regarding CPU and I/O costs with potential for parallelism. Complex similarity queries consisting of more than one similarity predicate have been studied considering complex similarity queries over a single feature [10] and over multiple features [7, 3]. Algorithms to perform a spatial selection and a spatial join simultaneously are presented in [14]. Indexing structures are used in [10, 3] to enhance complex similarity queries in relevance feedback environment. None of these works consider combining multiple similarity predicates through a set of operations, but instead they use intermediary scoring functions to evaluate the overall scores that indicate the pertinence of each element to the answer.

2.2 Query rewriting

A query involving complex Boolean expressions in RDBMS can be rewritten in an equivalent expression. Rewriting into Conjunctive or Disjunctive Normal Form (CNF or DNF) using one index per expression is a well-known technique [19]. Multiple indexes techniques are presented in [15]. Other approaches involve optimizing user-defined predicates with varying evaluation cost and selectivity regarding *AND* [8] and *OR* [12, 7] expressions. Factorization also helps rewriting Boolean expressions aiming optimizations [6].

Similarity query optimization over multimedia data started to receive attention, addressing the rewriting and optimization of ranked queries using expensive predicates [5]. However, no work addressed optimizations based on query rewriting for the similarity-based select operators regarding complex expressions, which is the objective of this paper.

3. BASIC OPERATORS AND VARIATIONS FOR SIMILARITY QUERIES

Aiming at covering as many query options as possible, we consider that each type of basic similarity predicate corresponds to as a basic, but flexible operator. Thus, a range predicate $Rq(s_q, r_q)$ corresponds to a $Range(\theta, s_q, r_q)$ operator, and a k-NN predicate $kNN(s_q)$ corresponds to a $Nearest(\theta, s_q, k)$ operator. The parameter θ is one of the relational operators $<, \leq, >, \geq, =$ or $\neq$. The basic range and k-NN predicate uses '$\leq$' as θ in the corresponding operators.

There are variations of each similarity predicate, and most of them can be expressed using the two basic operators. A common variation is obtaining the farthest instead of the nearest elements. Thus, a reversed range predicate $Rq^{-1}(s_q, r_q)$ asks for the elements s_i in the data set S such that $d(s_i, s_q) > r_q$. This is obtained when the θ parameter in the range operator is $>$. A k-farthest neighbor predicate $kFN(s_q)$ is equivalent.

Table 1 lists the basic queries and the variations used in this paper, together with their predicates and the corresponding operators. In this paper we use only $\leq$ and $>$ as θ for both operators. Table 2 lists the symbols used in this paper.

Table 1: Basic Predicates and Operators

Query Name	Predicate	Operator
Range Query	$Rq(s_q, r_q)$	$Range(\text{'}\leq\text{'}, s_q, r_q)$
Reversed Range Query	$Rq^{-1}(s_q, r_q)$	$Range(\text{'}>\text{'}, s_q, r_q)$
k-Nearest N. Query	$kNN(s_q)$	$Nearest(\text{'}\leq\text{'}, s_q, k)$
k-Farthest N. Query	$kFN(s_q)$	$Nearest(\text{'}>\text{'}, s_q, k)$

Table 2: Table of Symbols

Symbols	Definitions
$\mathbb{S}$	Set of all valid elements in the data domain.
S	Data set where queries are posed. $S \subset \mathbb{S}$
ν	Number of elements in data set S. $\nu = \|S\|$
$d(s_i, s_j)$	Distance function, or dissimilarity function. $d : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}^+$, $s_i, s_j \in \mathbb{S}$
s_q	Query center. $s_q \in \mathbb{S}$
r_q	Query radius. $r_q \geq 0$
k	Maximum number of elements in a query. $k \geq 1$
p_i, q_i	Predicate type p, where p is either Rq or kNN, and q is the other predicate type.
u_i, v_i	Limiting value of predicate p_i or q_i. If $p_i = Rq(s_q, r_q) \Rightarrow u_i = r_q$, else $u_i = k$.
m_i^p, m_i^q	Conjunction of predicates of one type. Used in SCSO expressions. $m_i^p = \neg p_k \wedge p_j$
m_i^{pq}	Conjunction involving Rq and kNN. Used in SCMO expressions. $m_i^{pq} = m_i^p \wedge m_i^q$
${}^c m_i^p$	Conjunction of pred. centered at object c Used in MCMO expressions.
m_1^p	The min-term consisting of the single non-complemented predicate. $m_1^p = p_j$
m_0^p	The min-term consisting of the single complemented predicate. $m_0^p = \neg p_i$

4. COMPLEX SIMILARITY QUERIES WITH A SINGLE CENTER

We define complex similarity queries as those composed of two or more basic similarity predicates combined by the Boolean operators $\wedge$ (and), $\vee$ (or) and $\neg$ (not). To analyze expressions representing complex similarity queries we divide the expressions into those having every predicate centered at the same query object (centers), and those having distinct centers. The former expressions are further divided into those composed of only a single similarity operator, and those involving both range and k-NN operators. Single operator means either only range or only k-NN operators, resulting in three classes:

- SCSO - single center/single operator;
- SCMO - single center/multiple operator; and
- MCMO - multiple centers/multiple operator.

In this section we analyze the expressions combining similarity operators over a single center $s_q \in \mathbb{S}$ applied over the data set $S \subset \mathbb{S}$ of a metric domain $\mathbb{S}$ with cardinality $\nu = |S|$.

4.1 Single center and same operator

We present in Appendix six basic properties of similarity predicate combination, and show that both the fields of range and of k-NN predicates centered at the same element forms corresponding Boolean Algebras. Therefore, any SCSO expression always can be expressed in disjunctive or conjunctive normal form (DNF or CNF). Both DNF and CNF share dual properties, so without loss of generality we consider only the DNF in the following discussion. A DNF expression consists of a disjunction of conjunctive min-terms, where each conjunctive min-term is the conjunction of either one variable or its complement. The variables are the basic similarity predicates. Therefore, any SCSO expression can always be represented in DNF.

To simplify the notation for SCSO expressions, let $p_i, i = 1...n$ be a predicate using a basic similarity operator p, where p is either Range or Nearest. Let also u_i be the limiting value of predicate p_i, so if p is Range then u_i is the radius r_i of the predicate $Rq(s_q, r_i)$, otherwise u_i is the number of neighbor k_i of the predicate $k_iNN(s_q)$. Let us also express a conjunction of predicates on p as m_a^p. Therefore, a SCSO expression involving predicates p_i on the similarity operator p and the $\wedge$, $\vee$ and $\neg$ Boolean operators can always be expressed in DNF as $(\neg p_1 \wedge \ldots \neg p_i \wedge p_j \wedge \ldots p_k) \vee (\neg p_x \wedge \ldots p_y) \vee \ldots = m_a^p \vee m_b^p \vee \ldots$. Notice that if p_i is a range predicate $p_i = Rq(s_q, r_i)$ then $\neg p_i = Rq^{-1}(s_q, r_i)$, and if p_i is a k-NN predicate $p_i = k_iNN(s_q)$ then $\neg p_i = (\nu - k_i)FN(s_q)$.

The following three Theorems apply to conjunctive min-terms, and can be straightforwardly demonstrated using the properties presented in the Appendix. Property 3 on Range operators and Property 5 on Nearest operators allow simplifying min-terms of the form $m_a^p = p_1 \wedge p_2 \wedge \ldots p_n$. Here we give an intuition of the theorems using Figure 1, which shows a two-dimensional space with the Euclidean DF, so the points nearer than a limiting radius r to a center point s_q are within a circle of radius r centered at s_q.

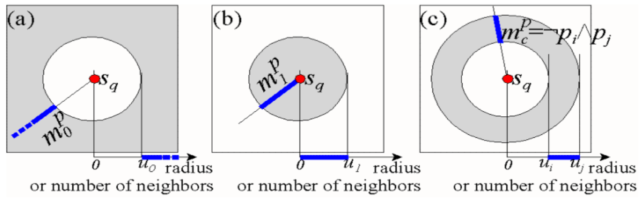


Figure 1: Regions covered by min-terms considering an Euclidean distance function in a 2D space. (a) min-term m_0^p; (b) min-term m_1^p; (c) min-term m_c^p.

Theorem 1: *Two basic predicates per Min-terms* - Each min-term in a SCSO DNF expression is composed of at most two basic predicates, one complemented and the other not complemented:

$$\widetilde{\sigma}_{\left((p_1 \wedge \dots p_g) \wedge (\neg p_h \wedge \dots \neg p_k)\right)} S \Leftrightarrow \widetilde{\sigma}_{(p_i \wedge \neg p_j)} S \ ,$$

where $u_i = min(u_1, \dots u_g)$ and $u_j = max(u_h, \dots u_k)$.

The min-terms having only one basic predicate are either the one not having a complemented predicate, which we represent as m_1^p, or the one having only the complemented predicate, which we represent as m_0^p. Min-terms m_0^p and m_1^p in a two-dimensional Euclidean domain are shown in Figure 1(a) and (b) respectively.

Theorem 2: *Min-terms as Rings* - If the limiting value of the complemented predicate of a min-term m_c^p with two basic predicates is less or equal than the limiting value of the non-complemented predicate, then m_c^p defines a ring-shaped region, otherwise it can be dropped from the DNF expression. That is:

$$\widetilde{\sigma}_{\left((\neg p_i \wedge p_j) | u_i > u_j\right)} S \Leftrightarrow null \ .$$

Figure 1(c) shows a two-predicate min-term m_c^p. DNF expressions combine one- and two-predicate min-terms leading to complex expressions, as the one shown in Figure 2.

Theorem 3: *Overlapping Min-terms* - Min-terms with overlapping limits can be joined as follows. (*total overlapping:*)

$$\widetilde{\sigma}_{\left((\neg p_g \wedge p_j) \vee (\neg p_h \wedge p_i) | u_g \le u_h \le u_i \le u_j\right)} S \Leftrightarrow \widetilde{\sigma}_{\left(\neg p_g \wedge p_j\right)} S \ ,$$

(*partial overlapping:*)

$$\widetilde{\sigma}_{\left((\neg p_g \wedge p_i) \vee (\neg p_h \wedge p_j) | u_g \le u_h \le u_i \le u_j\right)} S \Leftrightarrow \widetilde{\sigma}_{\left(\neg p_g \wedge p_j\right)} S \ .$$

Figure 2 shows the min-term $m_c^p = \neg p_4 \wedge p_7$ totally overlapping min-term $m_b^p = \neg p_5 \wedge p_6$, therefore only the min-term m_c^p must be retained. Likewise, the min-term $m_d^p = \neg p_9 \wedge p_{11}$ partially overlap min-term $m_e^p = \neg p_8 \wedge p_{10}$, therefore they can be joined in the single min-term $m_d^p \vee m_e^p = \neg p_8 \wedge p_{11}$. Min-term m_0^p can be checked for overlap considering that the missing predicate is p_∞ with $u_\infty = \infty$, and the min-term m_1^p checked using p_0 with $u_0 = 0$.

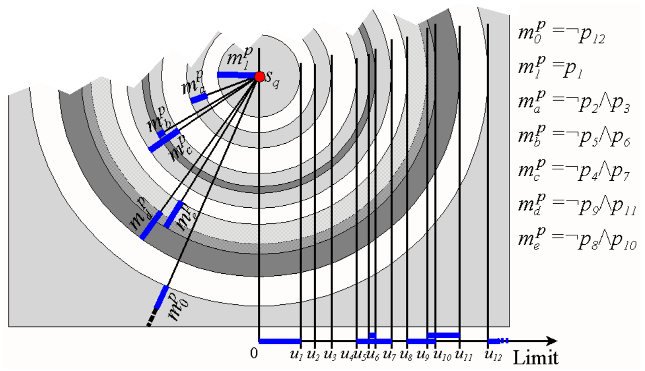

m_1^p
s_q
m_a^p
m_b^p
m_c^p
m_d^p
m_e^p
m_0^p
$m_0^p = \neg p_{12}$
$m_1^p = p_1$
$m_a^p = \neg p_2 \wedge p_3$
$m_b^p = \neg p_5 \wedge p_6$
$m_c^p = \neg p_4 \wedge p_7$
$m_d^p = \neg p_9 \wedge p_{11}$
$m_e^p = \neg p_8 \wedge p_{10}$
0
u_1 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 u_{10} u_{11} u_{12}
Limit

Figure 2: Regions involved in the disjunctive normal form $p_1 \vee (\neg p_2 \wedge p_3) \vee (\neg p_5 \wedge p_6) \vee (\neg p_4 \wedge p_7) \vee (\neg p_8 \wedge p_{10}) \vee (\neg p_9 \wedge p_{11}) \vee \neg p_{12}$, considering range queries and the Euclidean distance function in a 2D space.

Operators to execute SCSO expressions

Single Center/Single Operator expressions in DNF simplified by Theorems 1 to 3 cannot be further simplified, so a DBMS supporting similarity queries must be able to execute a simplified expression. There are two approaches: an algorithm for each basic operator processing a min-term at a time, or an algorithm executing the full expression at once.

The first approach employs the existing algorithms to execute the $Range(\theta, s_q, r_q)$ and $Nearest(\theta, s_q, k)$ basic predicates of min-terms m_1^p and m_0^p. However, two other algorithms must evaluate min-terms composed of two basic predicates: the $RingRange(s_q, r_{qi}, r_{qe})$ and the $RingNearest(s_q, k_{qi}, k_{qe})$, each one executing a min-term m_i^p over the corresponding operator. These algorithms can be created modifying the corresponding basic algorithms as follows. The $RingRange(s_q, r_{qi}, r_{qe})$ algorithm changes the comparison of an element s_i meeting the predicate $d(s_i, s_q)\theta r_q$ in the basic algorithm to the predicate $r_{qi} < d(s_i, s_q) \le r_{qe}$ in the ring range algorithm. The $RingNearest(s_q, k_{qi}, k_{qe})$ algorithm performs just the $Nearest(\le, s_q, k_{qe})$ operation, dropping the k_{qi} nearest neighbors from the final answer. The simplification process guarantees that each region covered by a min-term does not overlap any other, thus a complete SCSO expression can be evaluated calling a ring operator once for each min-term and concatenating the partial results to get the final answer.

The second approach to execute SCSO expressions employs a generalized algorithm to evaluate the full expression at once. We define the $GenericRange(s_q, rLimit[])$ algorithm to process range predicates, where $rLimit[]$ is an array of pairs $< r_i, r_e >$ with increasing radii, each one corresponding to a min-term. It is similar to the $RingRange()$ algorithm, but the predicate compares each element s_i to check if it is inside a valid region defined in $rLimit[]$. An equivalent $GenericNearest(s_q, kLimit[])$ algorithm shown as Algorithm 1 is defined for k-NN predicates. It first sorts every element based on its distance to the predicate center, then includes in the answer those elements whose rank is inside the extent defined by a min-term given by the pairs $< k_i, k_e >$ in $kLimit[]$.

It is not worth to use the first approach in expressions composed of several min-terms with k-NN predicates, because the rank of each element regarding its proximity to a predicate center cannot be determined without knowing the other elements in the set. Therefore, answering SCSO queries with on k-NN predicates imposes the second approach. The cost of executing the $GenericNearest()$ algorithm is equivalent to a single execution of the $Nearest()$ using the largest k in $kLimit[]$. Therefore the $GenericNearest()$ algorithm is always faster than calling the $Nearest()$ algorithm several times, whenever the number of limits $MaxK$ exceeds one.

Algorithm 1 $GenericNearest(s_q, kLimit[])$

```
1: $CoverObjSet = Nearest(s_q, kLimit[MaxK].k_e)$
2: for each $j$ in $CoverObjSet$ do
3:     if $j.rank()$ is in $kLimit$ then
4:         $Answer.Add(CoverObjSet[j])$
```

4.2 Single center and multiple operator type

To analyze a SCMO expression, we extend the p_i/u_i notation of SCSO expressions to represent SCMO expressions in the following way. If p_i is a range predicate with limiting radius u_i then q_j is a k-NN predicate with limiting $k = v_i$. Similar to m_a^p, m_b^q represents a disjunctive min-term of predicates of type q. The symbols p and q can be arbitrarily assigned to either range or nearest operators, provided they are distinct from each other. A conjunctive predicate $p_i \wedge q_j$ on a single center s_q recovers the elements satisfying both basic predicates. Therefore, if $p_i \wedge q_j = k_i NN(s_q) \wedge Rq(s_q, r_j)$ then $\widetilde{\sigma}_{\left(p_i \wedge q_j\right)} S = \widetilde{\sigma}_{\left(k_i NN(s_q)\right)} S \cap \widetilde{\sigma}_{\left(Rq(s_q, r_j)\right)} S$, leading to the following Theorem.

Theorem 4: *The field of range* ***and*** *k-NN predicates with same center forms a Boolean Algebra* - The conjunction of a k-NN predicate with a range predicate satisfies the commutative, associative and distributive properties over both $\wedge$ and $\vee$ Boolean operators.

The importance of this theorem relies on representing every single-centered expression as a SCMO expression in DNF, where each min-term is a conjunction of range and k-NN predicates, complemented or not. Moreover, as each operator range or nearest can contribute with at most two predicates, each min-term is a conjunction of at most four predicates: a reversed range, a k-farthest neighbor, a range and a k-NN. As these predicates are commutative over $\wedge$, each min-term of a DNF expression always can be expressed as $m_a^{pq} = (m_b^p \wedge m_c^q) = (\neg p_g \wedge \neg q_h) \wedge (p_i \wedge q_j)$.

Any conjunction $Rq(s_q, r_q) \wedge kNN(s_q)$ or $Rq^{-1}(s_q, r_q) \wedge kFN(s_q)$ of predicates centered at the same element s_q requires intersecting the intermediate results obtained by the basic operators. We propose a new $kAndRange(\theta, s_q, k, r_q)$ algorithm that receives the limits from both the range and the k-NN predicates and returns the elements that satisfy both criteria, so the following theorem can be stated.

Theorem 5: *Conjunction of k-NN* ***and*** *Range with same center* - The conjunction of a k-NN and a range predicate over dataset S is equivalent to the intersection of the results from both basic operators and equivalent to the execution of the algorithm $kAndRange(\theta, s_q, k, r_q)$. Therefore:

$$\widetilde{\sigma}_{\left(Rq(s_q,r_q)\right)} S \cap \widetilde{\sigma}_{\left(kNN(s_q)\right)} S \Leftrightarrow \widetilde{\sigma}_{\left(Rq(s_q,r_q) \wedge kNN(s_q)\right)} S \Leftrightarrow \widetilde{\sigma}_{\left(kAndRange(\theta,s_q,k,r_q)\right)} S \ .$$

Using the property of commutativity to represent every min-term m_a^{pq} as $(\neg p_g \wedge p_i) \wedge (\neg q_h \wedge q_j)$ it becomes the conjunction of a sub-expression on range with a sub-expression on k-NN predicates, and each one can be simplified using Theorems 1 to 3. Once simplified, the remaining min-terms can be re-arranged as $m_a^{pq} = (\neg p_g \wedge \neg q_h) \wedge (p_i \wedge q_j)$, and answered calling algorithm $kAndRange(\theta, s_q, k, r_q)$ twice, first to process $(\neg p_g \wedge \neg q_h)$ and then to process $(p_i \wedge q_j)$.

The intersection of two distinct min-terms in an SCMO expression in DNF can be non-empty. An algorithm to process such expressions can be improved guaranteeing that the min-terms are disjoint, so the following theorem is useful.

Theorem 6: *Disjoint min-terms in DNF expression* - Suppose a SCMO expression in DNF with at least two min-terms $m_a^{pq} = \big((\neg p_g \wedge \neg q_h) \wedge (p_i \wedge q_j)\big)$ and $m_b^{pq} = \big((\neg p_e \wedge \neg q_f) \wedge (p_k \wedge q_l)\big)$. If the limiting values $u_g \leq u_e \leq u_i$, then the min-term m_a^{pq} can be substituted by two non-overlapping min-terms:

$$m_a^{pq} = \big((\neg p_g \wedge \neg q_h) \wedge (p_e \wedge q_j)\big) \vee \big((\neg p_e \wedge \neg q_h) \wedge (p_i \wedge q_j)\big),$$

If the limiting values $u_e \leq u_i \leq u_k$ then the min-term m_b^{pq} can be substituted by two non-overlapping min-terms:

$$m_b^{pq} = \big((\neg p_e \wedge \neg q_f) \wedge (p_i \wedge q_l)\big) \vee \big((\neg p_i \wedge \neg q_f) \wedge (p_k \wedge q_l)\big).$$

When no pair of min-terms of a SCMO expression allows any of those substitutions, then each min-term is disjointed from any other min-term, so we call it an overlap-free, or a O_f-expression.

The substitutions in Theorem 6 can be applied to both predicate types, but it is enough to apply to just one. Notice that it fosters the use of incremental algorithms, and enables the use of the concatenation operation in place of the more expensive union operation.

Operators to execute SCMO expressions

SCMO expressions in DNF benefit from specific retrieval algorithms. A common sub-expression, worth to be implemented as a specific algorithm is the 'k-nearest and range query', defined as follows.

Definition 1: *k-Nearest and Range Query* - Given an element $s_q \in S$, a nearest value k, a distance $r_q \in \mathbb{R}^+$ and a relational operator $\theta \in \{\leq, >\}$, a 'k-Nearest and Range query' retrieves every element $s_i \in S \mid d(s_q, s_i) \theta r_q$ and $s_i \in Nearest(\theta, s_q, k)$.

A 'k-nearest and range query' can be executed by a $kAndRange(\theta, s_q, k, r_q)$ algorithm (see Algorithm 2). It executes the conjunction $p_i \wedge q_j$ of a range and a k-NN predicates centered at the same element, where p_i and q_j are both complemented or both not complemented. The conjunction requires both predicates satisfying the same θ condition, so this algorithm is based on the $Nearest(\theta, s_q, k)$ one. The allowed distances of the answers to the query center, represented by r_c, starts at r_q and is reduced (increased) as nearer (farther) elements are found. The answer is sorted by the distance of the element to the query center. The methods $Add()$ inserts a new element to the list keeping it sorted, $Length()$ returns the number of elements in the list, $DropLast()$ removes the farthest (if $\theta =$'$\leq$') or nearest (if $\theta =$'$>$') element in the list, and $MaxDist()$ returns the distance to the farthest or to the nearest (regarding θ) element in the list.

Algorithm 2 $kAndRange(\theta, s_q, k, r_q)$

```
1: set $Answer$ to $null$, set $r_c$ to $r_q$
2: for each $s_i$ in $S$ do
3:    Compute $d(s_i, s_q)$
4:    if $d(s_i, s_q)\ \theta\ r_c$ then
5:       $Answer.Add(s_i,\ \theta)$
6:       if $\neg(Answer.Length()\ \theta\ k)$ then
7:          $Answer.DropLast(\theta)$
8:          set $r_c$ to $Answer.MaxDist(\theta)$
9: return $Answer$
```

The $UniSimDNF(minterms[], s_q)$ algorithm (see Algorithm 3) answers a whole O_f-expression, using the $kAndRange(\theta, s_q, k, r_q)$ algorithm to solve each min-term. The intermediary results of each call to $kAndRange()$ is a set of elements sorted by their distances to the predicate center s_q. The final answer of a min-term is the intersection of both intermediary answer but, as both are sorted by the distance of each element to s_q, this operation can be computed with linear computational complexity on the number of elements involved.

The SCMO expression in DNF is sent to $UniSimDNF()$ as parameter $minterms[]$. As each min-term is calculated, the result is maintained in a queue sorted by the distance of each element to the query center s_q. Therefore, the union of each min-term to the previous ones can also be executed in linear time regarding the number of elements involved.

Algorithm 3 $UniSimDNF(minterms[], s_q)$

```
1: set Answer to null
2: for each $m_a^{pq} = (\neg p_g \wedge \neg q_h) \wedge (p_i \wedge q_j) \in minterms[]$ do
3:     set Answer to $Answer \bigcup \big(kAndRange(>, s_q, k_h, r_g) \bigcap kAndRange(\leq, s_q, k_j, r_i)\big)$
4: return Answer
```

5. COMPLEX SIMILARITY QUERIES WITH MULTIPLE CENTERS

Expressions involving the basic similarity predicates can include predicates centered at several elements. The complements of basic predicates do not depend on other predicates existing in the same expression, so Properties 1 and 2 also holds for MCMO expressions. The commutative, associative and distributive properties over the $\wedge$ and $\vee$ Boolean operators also hold for sub-expressions composed of the predicates centered at any element. Therefore, the field of MCMO expressions also forms a Boolean Algebra.

MCMO expressions can be represented in DNF with minterms composed of range or k-NN predicates, complemented or not, centered at the same or at distinct elements. MCMO expressions do not present special properties enabling further optimizations. However, we developed a technique to rewrite MCMO expressions that splits existing overlapping min-terms into other disjoint min-terms, exploiting the proposed properties of the SCSO and SCMO expressions to build efficient algorithms to answer MCMO queries.

For this discussion, suppose a MCMO query expression involving predicates centered at nc different elements $c_1, c_2, \ldots c_{nc}$, and let ${}^{c_k}m_a^{pq}$ represent a disjunction on both operators p and q centered at c_k, that is, ${}^{c_k}m_a^{pq} = (\neg p_g \wedge \neg q_h) \wedge (p_i \wedge q_j)$, where p_g, q_h, p_i and q_j are predicates centered at element c_k. Then, each min-term has the form: ${}^{c_1}m_a^{pq} \wedge {}^{c_2}m_b^{pq} \wedge \ldots {}^{c_{nc}}m_z^{pq}$.

To evaluate an MCMO query in DNF, our algorithm factorizes the query expression using one center at a time. Let us assume that the set of centers are maintained in a given order, so that c_i precedes c_j for every $i < j$. An expression factorized by center c_1 becomes $({}^{c_1}m_a^{pq} \wedge E_{a,1}) \vee ({}^{c_1}m_b^{pq} \wedge E_{b,1}) \ldots$ where $E_{i,j}$ is the i-th sub-expression in DNF involving predicates centered at elements $c_{j+1}, \ldots c_{nc}$. For example, the expression with two centers c_1, c_2:

$$({}^{c_1}m_a^{pq} \wedge {}^{c_2}m_b^{pq}) \vee ({}^{c_1}m_c^{pq} \wedge {}^{c_2}m_d^{pq}) \vee ({}^{c_1}m_a^{pq} \wedge {}^{c_2}m_e^{pq})$$

is factorized considering the center c_1 to

$$\big({}^{c_1}m_a^{pq} \wedge ({}^{c_2}m_b^{pq} \vee {}^{c_2}m_e^{pq})\big) \vee ({}^{c_1}m_c^{pq} \wedge {}^{c_2}m_d^{pq}).$$

Each $E_{i,j}$ is in turn recursively factorized considering one of the remaining centers. Each $E_{i,j}$ sub-expression and the full expression are simplified using Properties 1 to 3. After every sub-expression and the full expression had been simplified, they can be executed. The simplification process assures that each min-term in an expression do not overlap with the other min-terms in the same expression. Therefore, the implicit union operation required to integrate the answers of each min-term is executed as a concatenation operation, with linear processing cost regarding the number of elements concatenated.

The algorithm to execute a factorized MCMO query called $GenSimDNF(minterms[], centers[])$ is detailed in Algorithm 4. The parameter $minterms[]$ is an array with the min-terms of the expression, and $centers[]$ is an array with the centers. The algorithm works as follows. If an expression has only one center c, it returns the result of executing $UniSimDNF(minterms[], c)$. Otherwise, it chooses one center as c, and factorizes the expression finding each distinct term ${}^{c}m_i^{pq}$ occurring in the set of min-terms. For each min-term ${}^{c}m_i^{pq}$, it factorizes the remaining terms and calls $GenSimDNF()$ recursively. The answers from each sub-expression are intersected with the answers from the ${}^{c}m_i^{pq}$ term, generating the result of each min-term, which are in turn concatenated to obtain the final result of the query.

Algorithm 4 $GenSimDNF(minterms[], centers[])$ - Execute a generic similarity query in disjunctive normal form.

```
1: set Answer to null
2: if $|centers[]| > 1$ then
3:     set $c = FactorOut(centers[])$
4:     for each ${}^{c}m_i^{pq} \in minterm[]$ do
5:         $Prepare(minterms[], {}^{c}m_i^{pq}, InnerMinterms[])$
6:         set Answer to $Answer \bigcup \big(GenSimDNF(InnerMinterms[], centers[] - c) \bigcap UniSimDNF({}^{c}m_i^{pq}, c)\big)$
7:     return Answer
8: else
9:     return $UniSimDNF(minterms[], centers[1])$
```

Algorithm $FactorOut(center[])$, called in step 3 of $GenSimDNF()$, chooses one center to factor out, thus defining the order of the centers. Algorithm $FactorOut()$ can use different heuristics based on the sizes of the query radii relative to the dataset diameter, and on the number k of elements required relative to the number of elements in the data set. This algorithm can benefit from optimizations regarding particularities from a given application domain, so this is a topic where further research can improve performance. To perform the experiments presented in the next section, we have used a generic heuristic that we found useful in a variety of application domains. This heuristic uses the centroid of the remaining centers, selected as follows. Calculate the summation of the squared distances from each center to every other, then select the center whose summation is the smallest. The algorithm $Prepare(minterms[], {}^{c}m_i^{pq}, InnerMinterms[])$ selects from $minterms[]$ those having the term ${}^{c}m_i^{pq}$, strips it off and stores the resultant sub-expression into $InnerMinterms[]$.

A factored expression can have more min-terms than the equivalent DNF expression, but it will never have more predicates p_i. Therefore, the factorized expression will never require more incursions into the dataset to retrieve the partial answers than the DNF expression. Similarity queries over multimedia data can be very expensive. However, in our approach the overhead of the factorization cost of the $GenSimDNF()$ algorithm is overcame by the expressive gain in the query execution, as shown in the next section.

6. EXPERIMENTAL RESULTS

This paper proposes rewriting techniques to improve processing similarity queries. In this section we present experiments comparing the proposed algorithms with compositions of the basic algorithms combined by set-theoretical operators (the traditional approach), and show that the proposed algorithms are much faster and scalable considering database size.

The algorithms were implemented in C++, and the experiments ran in an AMD Athlon XP 2600 processor with 385MB of main memory, under the Linux operating system. Every test was performed using both sequential scan (SeqScan) and a Slim-tree index. We present the results obtained from the following four data sets:

- *LBeach*: a set of 36,298 2-dimensional coordinates of the road intersections in Long Beach City, CA, from the TIGER system of the U.S. Bureau of Census, using the Euclidean distance;
- *XR*: a set of 40,000 Metric Histograms (non-dimensional) obtained from various human body part radiographies, using the *MH*() distance function (the Metric Histogram is a piecewise linear approximation built over normalized histograms. As the number of pieces varies from an image to another, it does not have a defined dimension. The *MH*() distance uses the valleys and peaks to compare pairs of Metric Histograms [21]);
- *EngWords*: a random subset of 24,893 words from the English language, using the L_{Edit} distance;
- *Synt30D*: a synthetic set of 1,000,000 randomly generated 30-dimensional points, each coordinate in the range $[0,1]$, using the Euclidean distance.

Due to space limitations, in this paper we highlight the performance regarding only total time, as it summarizes the whole computational cost. Each measured point in the graphs represents the total time in seconds (log scale) to evaluate 200 queries with constant values for r and k and different centers.

6.1 Performance experiments

We evaluated the proposed algorithms comparing them with the traditional ones to query the real world data sets. The first experiment evaluates the time required to execute a complex similarity query on a pure metric data set (*EngWords*), processing the SCMO expression corresponding to Query **Q1** stated in Section 3:

$$\tilde{\sigma}_{\left(5NN(word_i)\right)} EngWords \cap \tilde{\sigma}_{\left(Rq(word_i,r)\right)} EngWords \ .$$

Figure 3 shows the total time required to process 200 queries asking for the $k = 5$ most similar words differing not more than r letters from a query center, for r varying from 1 to 10. The query centers are words randomly chosen in the data set. Plots A and C show the total time of the proposed *kAndRange*() algorithm respectively using Slim-tree and sequential scan. Plots B and D show total time using the traditional approach running both algorithms consecutively, using Slim-tree and sequential scan respectively. When using the L_{Edit}, the number of words retrieved rapidly increases as r increases, and for $r > 8$ it is retrieved, in average, more than 90% of the data set. The intersection operation required by the conjunction of the traditional algorithms can be quadratic on the cardinality of the sets intersected, explaining way plots B and D in Figure 3 present noticeable increasing starting on $r \approx 5$, whereas for smaller values

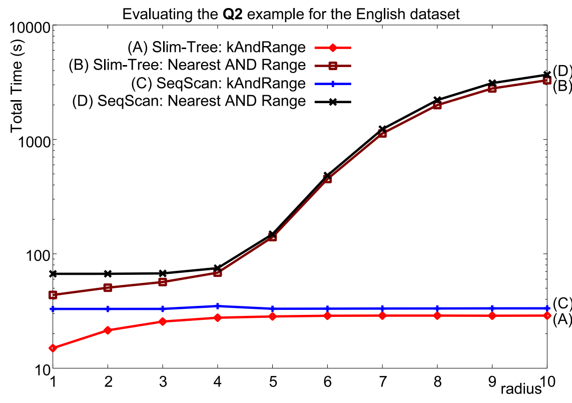


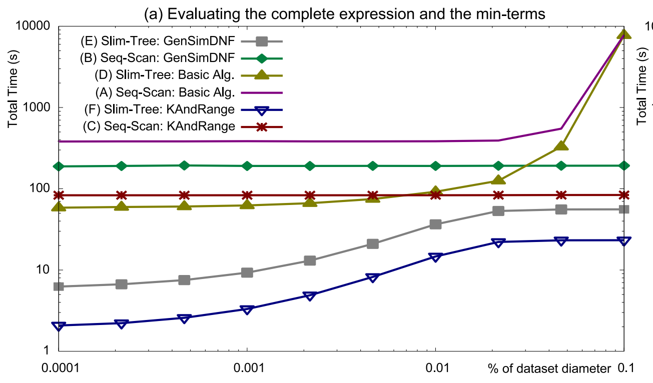
Figure 3: Results of executing the following query over the *EngWords* data set, using the L_{Edit} DF: $\tilde{\sigma}_{\left(5NN(word_i)\right)} EngWords \cap \tilde{\sigma}_{\left(Rq(word_i,r)\right)} EngWords$.

of r the time is mostly spent in the retrieval operations. However, the proposed *kAndRange*() algorithm does not suffer from this problem (as seen in plots A and C). Therefore, the proposed techniques lead to speedups from at least two times faster (sequential scan for radius from 1 to 3) to more than a hundred times faster (large radius for any access method). Moreover, even when the indexing structure reflects the explosion of words for larger values of r (plot A), the $k = 5$ limit always guarantees a performance better than sequential scan, even for large values of r (plot C).

The second experiment evaluates the execution time of the MCMO expression corresponding to Query **Q3** of Section 3 over *XR*, a pure metric data set. Figure 4 shows the results of asking for the 10 images most similar to each of three images e_1, e_2 and e_3 but not exceeding a given radius r. The radius r varies from 0.01% to 10% of the data set diameter, and the number of nearest neighbors is fixed at $k = 10$. As in the previous experiment, each measurement represents the total time in seconds to evaluate 200 queries for the same r and k and three randomly chosen centers.

Figure 4(a) shows the results of evaluating the complete query in the traditional way (plots A and D), and optimizing the query with the proposed algorithms (plots B and E). It also shows the time to execute just **one** min-term through the proposed *kAndRange*() algorithm (plots C and F), since the processing of each min-term gives equivalent measurements. Figure 4(b) shows the time to obtain intermediary results, highlighting where the answering process spends more time. It shows the time to evaluate one range predicate (the $\tilde{\sigma}_{\left(Rq(e_1,r)\right)} XR$) (plots I and L), one k-NN predicate (the $\tilde{\sigma}_{\left(10NN(e_1)\right)} XR$) (plots H and K) and the results of executing one min-term both through the proposed *kAndRange*() algorithm (plots C and F) and through the intersection of the range and k-NN algorithms (plots G and J). Plots C and F appears in both Figures 4(a) and 4(b) for reference. All plots are in log-log scales.

As it can be seen in Figure 4(a), for smaller values of r the proposed method (plots C and F) consistently requires approximately one third of the time required by the basic algorithms (plots A and D). This reduction comes from the query expression having three min-terms. This figure confirms that the optimization-based algorithms can take into



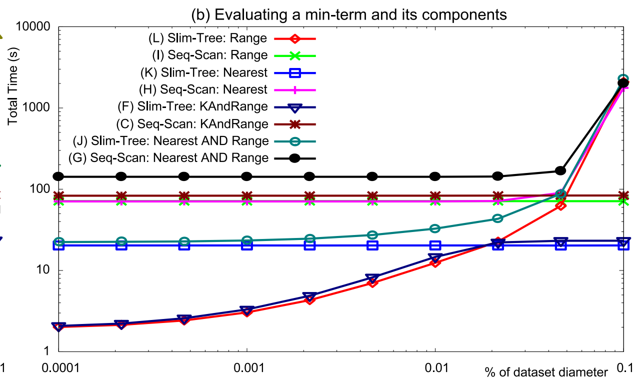


Figure 4: Results of executing the following query over a set of metric histograms of 40,000 x-Ray images of various parts of the human body, using the Metric-Histogram MH() metric: $\left(\widetilde{\sigma}_{\left(Rq(e_1,r)\right)} XR \cap \widetilde{\sigma}_{\left(10NN(e_1)\right)} XR\right) \cup \left(\widetilde{\sigma}_{\left(Rq(e_2,r)\right)} XR \cap \widetilde{\sigma}_{\left(10NN(e_2)\right)} XR\right) \cup \left(\widetilde{\sigma}_{\left(Rq(e_3,r)\right)} XR \cap \widetilde{\sigma}_{\left(10NN(e_3)\right)} XR\right)$.

account the other min-terms when processing one min-term, improving the overall performance.

Another important point shown in Figure 4(b) is that the *kAndRange*() algorithm improves the query execution by limiting the number of retrieved elements when using access methods. Observe that the plot corresponding to the range algorithm (plot L) increases continuously, surpassing the plot of the k-NN algorithm (plot K). However, as the number of elements retrieved by the range algorithm approaches k, the curve corresponding to the *kAndRange*() algorithm flattens (plot F), always remaining lower than the combination of the range and k-NN algorithm (plot J). This effect is reflected in the time spent to evaluate the complete query, so the improvement for large values of r is even more remarkable, as it can be seen comparing plot E with plot D of Figure 4(a). The pruning obtained by limiting the number of elements in the range part of the min-term also reduces the complexity of merging the intermediary results of each min-term. As it can be seen comparing plots E and D of Figure 4(a), the speedup obtained achieves more than 130 times for large radii and small k.

Query **Q3** is composed of three min-terms at different centers, and plots E and F of Figure 4(a) shows that the time required to evaluate the complete query by the proposed method is about one third of the time required by the basic algorithms. This is due to the cost of the *UniSimDNF*() algorithm being equivalent to a single call to *kAndRange*(), whereas the traditional approach needs calling the basic algorithms the same number of times as there are min-terms. Experimental evaluations confirmed that more complex queries lead to correspondingly larger improvements.

The third set of experiments evaluates the time required to execute the MCMO expression derived from Query **Q2** of Section 3 over the *LBeach* data set. Figure 5 shows the results of asking for the 10 closest road intersections not farther than radius r_1 from center mj and not farther than radius r_2 from center wj. Radii r_1 are the abscissas and r_2 is randomly chosen in $[0, r_1]$. Figure 5 shows the time required to evaluate the full expression using both the proposed and the traditional approaches, using the Slim-tree (plots A and C) and sequential scan (plots B and D). In this case, the gain for the proposed algorithm is about 35% for small radii, the ones most frequently asked in similarity queries.

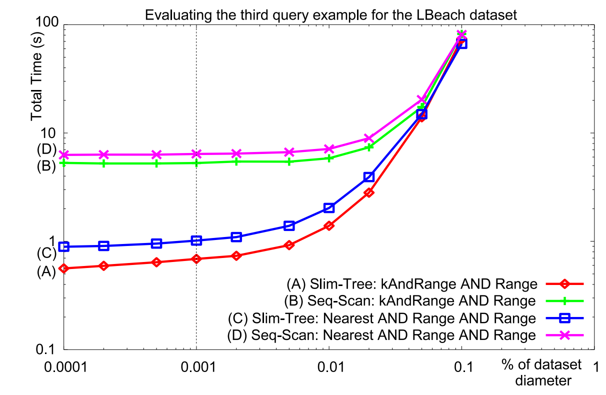


Figure 5: Evaluating Query Q2 of Section 3 over the *LBeach* data set using the Euclidean DF: $\left(\widetilde{\sigma}_{\left(10NN(mj)\right)} \textbf{\textit{LBeach}} \cap \widetilde{\sigma}_{\left(Rq(mj,r_1)\right)} \textbf{\textit{LBeach}}\right) \cap \widetilde{\sigma}_{\left(Rq(wj,r_2)\right)} \textbf{\textit{LBeach}}$.

6.2 Scalability

To evaluate the scalability of the proposed algorithms, we performed two experiments. The first employed the expression of Query **Q2** from Section 3 over the *LBeach* data set. For this experiment, we generated random samples of 5,000 elements, 10,000 elements, and so on from the *LBeach* data set, and measured the total time to calculate 200 queries using $k = 37$ (equivalent to 0.1% of the data set size) and $r_1 = 0.001$ of the data set diameter. The result, shown in Figure 6, indicates that the proposed algorithms have a linear behavior regarding the data set size.

The second scalability experiment evaluated the *kAndRange*() algorithm using the *Synt30D* data set, showing the total time to calculate 200 queries considering $k = 10$ and $r_q = 0.02$ (see Figure 7). Each set of 200 queries was performed increasing the data set in steps of 50,000 elements. The measurements show that the behavior of this algorithm is linear even for very large data sets, that do not fit in main memory, as it happens with the *Synt30D* data set.

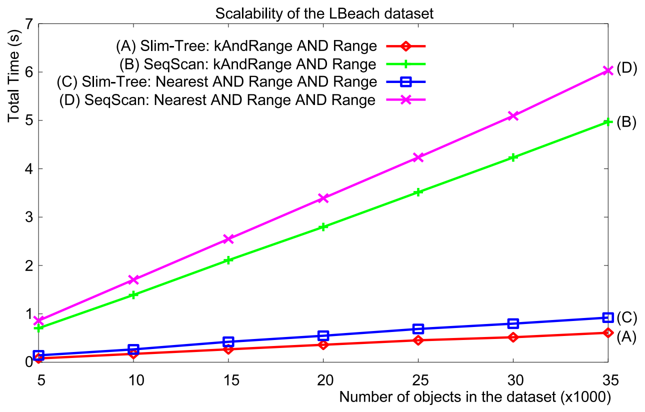


Figure 6: Scalability test using Query Q2 of Section 3 over the *LBeach* data set.

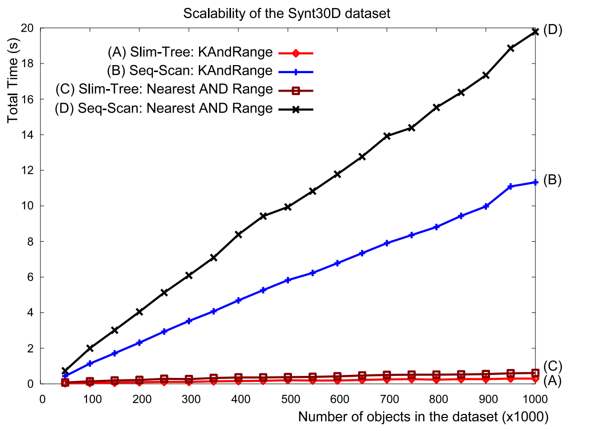


Figure 7: Scalability test regarding the *kAndRange*() algorithm over the *Synt30D* data set.

7. CONCLUSIONS

Similarity queries have been increasingly demanded to retrieve complex data in large data sets but, until now, there was no study on how to analyze and optimize a query expression involving more than one similarity predicate. This paper brings the following contributions to this subject:

1. It presents rules to derive formal optimizations for similarity queries, exploring the representation of the queries in disjunctive and conjunctive normal form. It pays special attention to DNF, which is the usual way to represent selections in traditional databases.

2. The algebraic approach adopted helps to identify which are the primitive operators required for a DBMS to support similarity queries, revealing that quite a few are really needed. The paper also presents the algorithms for the newly identified operators.

3. The rules developed for similarity query optimizations hold for spatial and metric datasets as well. They are independent of the underlying MAM employed, although specific implementations can take advantage of the index structure being used.

4. We implemented the prototype of a similarity query analyzer/executor, which showed that the optimization techniques proposed significantly improve query answering. Experiments on both real and synthetic datasets show that they accelerate answering similarity queries more than two orders of magnitude. Scalability experiments confirm that the techniques are scalable over both range and k-nearest neighbor queries, keeping the gains for any dataset size.

Throughout the paper we answered the three main issues stated in Section 1. Starting with the third issue, "*How to adequately represent a query to be submitted to multi-purpose similarity search algorithms*": it is solved by dividing the problem of analyzing complex similarity expressions in three steps – Single Center/Single Operator (SCSO), Single Center/Multiple Operators (SCMO), and Multiple Centers/Multiple Operators (MCMO). Following this approach, rules to simplify a complex similarity expression were straightforwardly obtained, solving the second issue: "*What rules guide the query rewriting process to explore strategies to execute similarity search algorithms*". Those rules allow representing the query expressions in a way such that union and intersection operations can be performed with linear computational cost regarding the number of elements involved, as opposed to the usual super-linear complexity required by traditional methods. Thereafter, we presented algebraic rules to optimize similarity queries, and a small collection of simple and generic similarity-based retrieval algorithms able to answer complex similarity queries in linear time, solving the first issue "*Which algorithms are efficient to answer queries composed of conjunctions and disjunctions of similarity predicates*". We identified that four algorithms are enough to answer any complex similarity query involving range and k-NN predicates and their discussed variations. They are the *GenericRange*(), the *GenericNearest*(), the *UniSimDNF*(), and the *GenSimDNF*() algorithms. Three others, the *RingRange*(), *RingNearest*() and *kAndRange*() were also included to improve performance when answering simpler but frequently asked queries.

Rules to simplify complex similarity expressions enable to generate multiple representations of a query. Selecting the one that leads to the best execution plan depends on a cost model, which is dependent from the underlying indexing structure employed. In this paper we concentrated on rules that are independent from indexing structures, so we do not elaborated on this subject. The experiments performed assume that the majority of the queries aim at retrieving few elements relative to the database cardinality. The results showed that the proposed approach is significantly better than the traditional one.

As a follow-up of this paper, we are working on the rules to extend the relational algebra to support similarity join and combinations of similarity and non-similarity based predicates. This will open the possibility to support the storage, relevance feedback and content-based retrieval of complex data, such as images, scientific and biological data, among others, in systems based on the relational algebra, such as the current RDBMS based on SQL.

Acknowledgement

This work has been supported by CNPq (Brazilian National Council for Research Supporting), FAPESP (São Paulo State Research Foundation) and CAPES/Fulbright (Brazilian Federal Agency for Post-Graduate Education).

8. REFERENCES

[1] S. Berchtold, C. Böhm, B. Braunmüller, D. A. Keim, and H.-P. Kriegel. Fast parallel similarity search in multimedia databases. In *ACM SIGMOD*, pages 1–12, 1997.

[2] C. Böhm, B. Braunmüller, and H.-P. Kriegel. The pruning power: Theory and heuristics for mining databases with multiple k-nearest-neighbor queries. In *Int. Conf. on DaWaK*, pages 244–257, 2000.

[3] K. Böhm, M. Mlivoncic, H.-J. Schek, and R. Weber. Fast evaluation techniques for complex similarity queries. In *VLDB*, pages 211–220, 2001.

[4] B. Braunmüller, M. Ester, H.-P. Kriegel, and J. Sander. Efficiently supporting multiple similarity queries for mining in metric databases. In *ICDE*, pages 256–267, 2000.

[5] K. C.-C. Chang and S. won Hwang. Minimal probing: supporting expensive predicates for top-k queries. In *ACM SIGMOD*, pages 346–357, 2002.

[6] S. Chaudhuri, P. Ganesan, and S. Sarawagi. Factorizing complex predicates in queries to exploit indexes. In *ACM SIGMOD*, pages 361–372, 2003.

[7] S. Chaudhuri and L. Gravano. Optimizing queries over multimedia repositories. In *ACM SIGMOD*, pages 91–102, 1996.

[8] S. Chaudhuri and K. Shim. Optimization of queries with user-defined predicates. *ACM TODS*, 24(2):177–228, 1999.

[9] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *VLDB*, pages 426–435, 1997.

[10] P. Ciaccia, M. Patella, and P. Zezula. Processing complex similarity queries with distance-based access methods. In *EDBT*, volume 1377, pages 9–23, 1998.

[11] G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM TODS*, 24(2):265–318, 1999.

[12] A. Kemper, G. Moerkotte, K. Peithner, and M. Steinbrunn. Optimizing disjunctive queries with expensive predicates. In *ACM SIGMOD*, pages 336–347, 1994.

[13] F. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel, and Z. Protopapas. Fast nearest neighbor search in medical image databases. In *VLDB*, pages 215–226, 1996.

[14] N. Mamoulis, D. Papadias, and D. Arkoumanis. Complex spatial query processing. *GeoInformatica Journal*, 8(4):311–346, 2004.

[15] C. Mohan, D. J. Haderle, Y. Wang, and J. M. Cheng. Single table access using multiple indexes: Optimization, execution, and concurrency control techniques. In *EDBT*, volume 416 of *LNCS*, pages 29–43. Springer, 1990.

[16] K. Porkaew, S. Mehrotra, and M. Ortega. Query reformulation for content based multimedia retrieval in MARS. *IEEE ICMCS*, 2:747–751, 1999.

[17] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *ACM SIGMOD*, pages 71–79, 1995.

[18] T. Seidl and H.-P. Kriegel. Optimal multi-step k-nearest neighbor search. In *ACM SIGMOD*, pages 154–165, 1998.

[19] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. In *ACM SIGMOD*, pages 23–34, 1979.

[20] M. Tasan and Z. M. zsoyoglu. Improvements in distance-based indexing. In *SSDBM*, pages 161–170, Greece, 2004.

[21] A. J. M. Traina, C. Traina Jr., J. Bueno, F. Chino, and P. Marques. Efficient content-based image retrieval through metric histograms. *WWW Journal*, 6(2):157–185, 2003.

[22] C. Traina Jr., A. J. M. Traina, C. Faloutsos, and B. Seeger. Fast indexing and visualization of metric datasets using Slim-trees. *IEEE TKDE*, 14(2):244–260, 2002.

APPENDIX

Basic Properties on Similarity Predicates

For the sake of completeness, we present in this Appendix the basic properties hold by SCSP expressions using only one basic similarity predicate with the ¬ operator. The following two properties govern the complement of range and k-NN predicates.

Property 1: *Range complement* - The complement of a range predicate is the reversed range predicate centered at the same element and same radius, so that:

$$\widetilde{\sigma}_{\left(\neg Rq(s_q,r_q)\right)} S \Leftrightarrow \widetilde{\sigma}_{\left(Rq^{-1}(s_q,r_q)\right)} S$$

Property 2: *k-NN complement* - The complement of a k-NN predicate is the $\bar{k}$-farthest neighbor predicate centered at the same element and with the number of required elements $\bar{k}$ equal to the total number of elements ν in data set S less k, so that:

$$\widetilde{\sigma}_{\left(\neg kNN(s_q)\right)} S \Leftrightarrow \widetilde{\sigma}_{\left((\nu-k)FN(s_q)\right)} S$$

Conjunctions and disjunctions of range predicates satisfy the properties following.

Property 3: (*Range* ***and*** *Range*) *with same center* - The conjunction of two range predicates is equivalent to a single range predicate using the smaller value between both radii as the predicate radius, so that:

$$\widetilde{\sigma}_{\left(Rq(s_q,r_{q1})\right)} S \cap \widetilde{\sigma}_{\left(Rq(s_q,r_{q2})\right)} S \Leftrightarrow$$
$$\widetilde{\sigma}_{\left(Rq(s_q,r_{q1}) \wedge Rq(s_q,r_{q2})\right)} S \Leftrightarrow \widetilde{\sigma}_{\left(Rq(s_q,min(r_{q1},r_{q2}))\right)} S$$

Combined with Property 1, the following also holds

$$\widetilde{\sigma}_{\left(Rq^{-1}(s_q,r_{q1})\right)} S \cap \widetilde{\sigma}_{\left(Rq^{-1}(s_q,r_{q2})\right)} S \Leftrightarrow$$
$$\widetilde{\sigma}_{\left(Rq^{-1}(s_q,r_{q1}) \wedge Rq^{-1}(s_q,r_{q2})\right)} S \Leftrightarrow \widetilde{\sigma}_{\left(Rq^{-1}(s_q,max(r_{q1},r_{q2}))\right)} S$$

Property 4: (*Range* ***or*** *Range*) *with same center* - The disjunction of two range predicates is equivalent to a single range predicate using the larger value between the radii of both basic predicates as the predicate radius. That is:

$$\widetilde{\sigma}_{\left(Rq(s_q,r_{q1})\right)} S \cup \widetilde{\sigma}_{\left(Rq(s_q,r_{q2})\right)} S \Leftrightarrow$$
$$\widetilde{\sigma}_{\left(Rq(s_q,r_{q1}) \vee Rq(s_q,r_{q2})\right)} S \Leftrightarrow \widetilde{\sigma}_{\left(Rq(s_q,max(r_{q1},r_{q2}))\right)} S$$

which, combined with Property 1, results in

$$\widetilde{\sigma}_{\left(Rq^{-1}(s_q,r_{q1})\right)} S \cup \widetilde{\sigma}_{\left(Rq^{-1}(s_q,r_{q2})\right)} S \Leftrightarrow$$
$$\widetilde{\sigma}_{\left(Rq^{-1}(s_q,r_{q1}) \vee Rq^{-1}(s_q,r_{q2})\right)} S \Leftrightarrow \widetilde{\sigma}_{\left(Rq^{-1}(s_q,min(r_{q1},r_{q2}))\right)} S$$

It is straightforward to show that range predicates follow the commutative, associative and distributive properties over the $\wedge$ and $\vee$ Boolean operators. Also, the range predicate with infinity radius is the identity element under the $\wedge$ operator. We define here a null predicate $\varnothing(s_q)$ that always returns the null set, so it is the identity element under the $\vee$ operator. In this way, the field of range predicates centered at the same element forms a Boolean Algebra.

Equivalent properties hold for the k-NN predicate as follows.

Property 5: (*Nearest* ***and*** *Nearest*) *with same center* - The conjunction of two k-NN predicates is equivalent to the one having the smallest k, that is:

$$\widetilde{\sigma}_{\left(k_1NN(s_q)\right)} S \cap \widetilde{\sigma}_{\left(k_2NN(s_q)\right)} S \Leftrightarrow$$
$$\widetilde{\sigma}_{\left(k_1NN(s_q) \wedge (k_2NN(s_q)\right)} S \Leftrightarrow \widetilde{\sigma}_{\left(min(k_1,k_2)NN(s_q)\right)} S$$

which, combined with Property 2, results in

$$\widetilde{\sigma}_{\left((\neg k_1NN(s_q)\right)} S \cap \widetilde{\sigma}_{\left(\neg k_2NN(s_q)\right)} S \Leftrightarrow$$
$$\widetilde{\sigma}_{\left((\nu-k_1)FN(s_q) \wedge (\nu-k2)FN(s_q)\right)} S \Leftrightarrow \widetilde{\sigma}_{\left(\nu-max(k_1,k_2)FN(s_q)\right)} S$$

Property 6: (*Nearest* ***or*** *Nearest*) *with same center* - The disjunction of two k-NN predicates is equivalent to the one having the largest k, that is:

$$\widetilde{\sigma}_{\left((k_1NN(s_q)\right)} S \cup \widetilde{\sigma}_{\left(k_2NN(s_q)\right)} S \Leftrightarrow$$
$$\widetilde{\sigma}_{\left(k_1NN(s_q) \vee (k_2NN(s_q)\right)} S \Leftrightarrow \widetilde{\sigma}_{\left(max(k_1,k_2)NN(s_q)\right)} S$$

which, combined with Property 2, results in

$$\widetilde{\sigma}_{\left((\neg k_1NN(s_q)\right)} S \cup \widetilde{\sigma}_{\left(\neg k_2NN(s_q)\right)} S \Leftrightarrow$$
$$\widetilde{\sigma}_{\left((\nu-k_1)FN(s_q) \vee (\nu-k_2)FN(s_q)\right)} S \Leftrightarrow \widetilde{\sigma}_{\left(\nu-min(k_1,k_2)FN(s_q)\right)} S$$

Like the range predicates, the k-NN predicates also follow the commutative, associative and distributive properties over the $\wedge$ and $\vee$ Boolean operators. Considering $\varnothing(s_q)$ as the identity element under $\vee$ operator, and the predicate requiring ν elements $\nu NN(s_q)$ as the identity element under $\wedge$ operator, the field of k-NN predicates centered at a single element also forms a Boolean Algebra.