Flexible Integration of Multimedia Sub-Queries with Qualitative Preferences

Ilaria Bartolini* and Paolo Ciaccia
({ibartolini,pciaccia}@deis.unibo.it)
DEIS, University of Bologna - IEIIT-BO/CNR, Bologna, Italy

Vincent Oria (oria@cis.njit.edu)
Dept. of Computer Science, NJ Inst. of Technology, Newark, NJ, USA

M. Tamer Özsu (tozsu@uwaterloo.ca)
School of Computer Science, University of Waterloo, Waterloo, ON, Canada

Abstract. Complex multimedia queries, aiming to retrieve from large databases those objects that best match the query specification, are usually processed by splitting them into a set of m simpler sub-queries, each dealing with only some of the query features. To determine which are the overall best-matching objects, a rule is then needed to integrate the results of such sub-queries, i.e., how to globally rank the m-dimensional vectors of matching degrees, or *partial scores*, that objects obtain on the m sub-queries. It is a fact that state-of-the-art approaches all adopt as integration rule a *scoring function*, such as weighted average, that aggregates the m partial scores into an overall (numerical) similarity score, so that objects can be linearly ordered and only the highest scored ones returned to the user. This choice however forces the system to compromise between the different sub-queries and can easily lead to miss relevant results.

In this paper we explore the potentialities of a more general approach, based on the use of *qualitative preferences*, able to define arbitrary *partial* (rather than only linear) orders on database objects, so that a larger flexibility is gained in shaping what the user is looking for. For the purpose of efficient evaluation, we propose two integration algorithms able to work with *any* (monotone) partial order (thus also with scoring functions): MPO, which delivers objects one layer of the partial order at a time, and iMPO, which can incrementally return one object at a time, thus also suitable for processing top k queries. Our analysis demonstrates that using qualitative preferences pays off. In particular, using *Skyline* and *Region-prioritized Skyline* preferences for queries on a real image database, we show that the results we get have a precision comparable to that obtainable using scoring functions, yet they are obtained much faster, saving up to about 70% database accesses.

1. Introduction

Specification and evaluation of multimedia (MM) queries are both difficult problems to be addressed for the development of effective MM tools and applications. Indeed, the formulation of a query on a MM database has to take into account both the intrinsic complexity to properly char-

* Part of this work was performed while this author was visiting NJIT.

acterize the semantic content of multimedia objects and the difficulty that a user experiences when trying to exactly formulate her needs. With a large MM database, in which each object is characterized by means of a set of relevant, automatically extracted, low-level *features* (e.g., color, texture, and shape in the case of still images), the user provides the system with a "target" (query) object and expects as result the "most similar" database objects. For this retrieval model to effectively work, it is well recognized that the similarity function used to compare objects has to be properly adapted, possibly by means of some *relevance feedback* technique [21], to fit the subjective user preferences.

When dealing with *complex* MM queries involving multiple features, the scenario is further complicated. Indeed, since it is a common case that features are separately indexed [19] or even managed by independent specialized sub-systems [14], an integration of partial results is needed. Relevant examples of integration algorithms are $\mathcal{A}_0$ [14], TA [16], and MEDRANK [15]. Their common rationale is to have an independent, yet synchronized, evaluation of sub-queries, one for each involved feature. Each object returned by a sub-query has an associated *partial score* for the corresponding feature, and such partial scores are then aggregated by means of some (possibly weighted) *scoring function*, like min and avg, into an *overall score*. Under this view one object is better than (i.e., preferred to/ranked higher than) another iff its overall score is higher.

It is well-known that the choice of the scoring function and of the weights are both critical factors for the determination of the final result, and that any choice necessarily provides a limited view of the best available alternatives (namely, only those maximizing the scoring function). Although relevance feedback mechanisms specifically proposed in the case of multiple sub-queries [20] can alleviate this problem by allowing the user to progressively shift her focus towards interesting regions of the search space, they usually require several iterations before leading to acceptable results, thus generating a not negligible overhead on the system [5]. Further, since scoring functions can only represent preferences that define a linear order on the objects [17], they have indeed a limited expressive power, which might severely limit their applicability in modern multimedia systems.

With the aim of going beyond the intrinsic limits of scoring functions, in this paper we propose a novel, more general, approach to subquery integration, based on *qualitative preferences*. Qualitative preferences, which have been recently found their way, among the others, in relational databases [11] and Web information systems [2], only require that, given a pair of objects o_i and o_j, one has some (binary) *preference relation* stating when o_i is preferred to o_j ($o_i \succ o_j$). This framework

clearly includes scoring functions as a special case (since for any scoring function $S()$ one can simply define $o_i \succ o_j$ iff $S(o_i) > S(o_j)$), yet it can also rely on more sophisticated and flexible criteria able to explicitly take into account all the partial scores when comparing objects. As a first beneficial effect this has the consequence that there is no risk of choosing "bad parameter values", as it might happen with scoring functions. In particular, using *Skyline* preferences [8], one obtains as result of a query all *Pareto-optimal* objects, i.e., all and only those objects that are maxima of some scoring function. Intuitively, this makes it possible to get an "overall view" of the potential best objects for a given query, a fact that highly simplifies the task of focusing on the part of the search space containing more relevant objects. Our experiments on a real-world image database indeed confirm that the results obtained using Skyline preferences cover much better than scoring functions the space of the relevant objects for a query, and that this effect is further amplified when using an original variant of the Skyline, called *Region-prioritized Skyline*.

The model of queries we consider includes the standard one, where one is interested in obtaining the top k results, the major difference being, of course, the criterion according to which objects are ranked. To this end we rely on the well-defined semantics of the *Best* operator [22], $\beta_{\succ}(C)$, that returns all the objects o in a collection C such that there is no object in C better than o according to $\succ$.[1] Ranking is naturally obtained by recursively applying the Best operator to the remaining objects (i.e., those in $C - \beta_{\succ}(C)$, and so on). This leads to a *layered* view of the search space where all the objects in one layer are equally ranked. Thus, besides top k queries, we also provide a "first ℓ layers" query model, which adds further flexibility to the retrieval phase.

Turning to consider evaluation issues, we propose two novel integration algorithms. Algorithm MPO applies to *any* preference relation that defines a strictly monotone partial order on database objects and works by returning one layer at a time of the partial order. In order to efficiently support also top k queries and to minimize the time a user has to wait to get any object in the result, we then introduce the iMPO incremental algorithm, which can deliver as earliest as possible objects one at a time.

Our experiments, contrasting Skyline (SL) and Region-prioritized Skyline (RS) preferences with the min and avg scoring functions, show that the quality of the results obtainable from both SL and RS, as measured in terms of classical *precision*, is comparable to that of averaging

[1] Operators with the same semantics of Best have been independently proposed in [11] and [18].

partial scores, whereas is (much) better than that of min; however, when using either SL or RS one needs to perform much less database accesses than with avg to get the same number of relevant objects. Further, with qualitative preferences one obtains results that much better reflect the actual distribution of the relevant objects of a query.

The rest of the paper is organized as follows. In Section 2 we provide the basic definitions concerning the query scenario and the integration problem; then we discuss scoring functions and their limits. Section 3 introduces qualitative preferences. In Section 4 we present the MPO and iMPO algorithms, and in Section 5 we describe experimental results. Table I lists relevant symbols used in the paper.

Table I. Relevant symbols used in the paper

Symbol	Description
C	collection of objects
o_i	the i-th object of C
$\mathcal{Q}$	complex query with m sub-queries
Q_q	q-th sub-query ($q = 1, \ldots, m$)
$s_{i,q}$	partial score of o_i with respect to Q_q ($s_{i,q} \in [0,1]$)
A	m-dimensional "answer space", $A = [0,1]^m$
$\mathbf{p_i}$	representative point of object o_i in the answer space
$\underline{\mathbf{p}}$	the "threshold point"
S	scoring function
s_i	overall score of object o_i computed by S, $s_i = S(\mathbf{p_i})$
$\succ$	preference relation
$\beta_{\succ}$	the Best operator

2. The Integration Problem

Consider a collection C of multimedia objects and a *complex* query, $\mathcal{Q} = (Q_1, Q_2, \ldots, Q_m)$, where each Q_q is a distinct sub-query. Note that this simple query model is powerful enough to include as relevant, cases where each Q_q refers to a distinct (subset of) feature(s) used in the query [21] and cases in which *local* features are used to characterize different parts of a same object. The latter cases are well exemplified by region-based image retrieval systems [4], in which each Q_q corresponds to a distinct region of the query image.

For each sub-query Q_q we assume that objects are assigned a *partial score*, $s_{i,q} \in [0,1]$, that tells us "how well" object o_i matches Q_q, with higher values being better. Consequently, for the purpose of determining the results of query $\mathcal{Q}$, object o_i can be univocally

represented by a point $\mathbf{p_i} = (s_{i,1}, \ldots, s_{i,m})$ in the m-dimensional *answer space* $A = [0,1]^m$, and the integration problem amounts to efficiently determine the "overall best" points (i.e., objects) in such space. Preliminary to the discussion of algorithmic issues is therefore a precise understanding of what "overall best" can actually mean.

2.1. Scoring Functions

According to the standard approach used to define the semantics of a complex multimedia query, partial scores are aggregated using some *scoring function* $S : A \rightarrow [0,1]$, that assigns to each point $\mathbf{p_i}$ in the answer space A an *overall score* $s_i = S(\mathbf{p_i})$.

In practice, commonly used scoring functions, like min, max, avg, etc., as well as their weighted versions, all satisfy the reasonable property of being *(strictly) monotone*, which guarantees that the overall score is always positively correlated with all the partial scores:

Definition 1 (Monotonicity of scoring functions) *A scoring function S is* monotone *if $s_{j,q} \leq s_{i,q}$ for all q implies $s_j = S(s_{j,1}, \ldots, s_{j,m}) \leq s_i = S(s_{i,1}, \ldots, s_{i,m})$, and* strictly monotone *if $s_{j,q} < s_{i,q}$ $\forall q$ implies $s_j < s_i$.*

Monotonicity and strict monotonicity can be given a simple yet useful geometric interpretation in the answer space A (see also Figure 1 (a)).

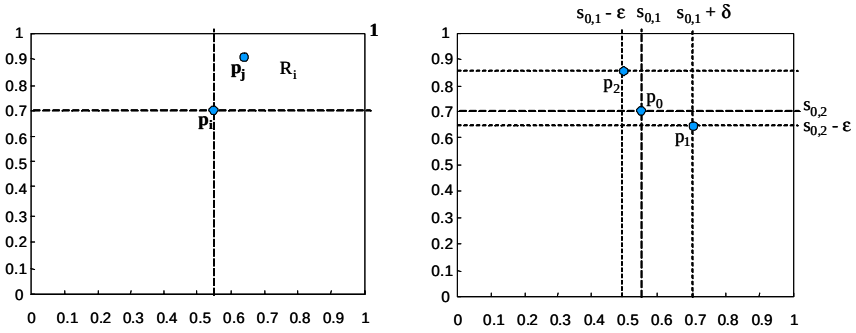


Figure 1. (a) The hyper-rectangle of object o_i; (b) Why the weighted average cannot retrieve all potential best matches

For this, consider the "ideal" result point $\mathbf{1} = (1, \ldots, 1)$, which corresponds to the best possible evaluation for all the sub-queries, and let R_i be the "hyper-rectangle of o_i" having $\mathbf{p_i}$ and $\mathbf{1}$ as opposite vertices. If S is (strictly) monotone and $\mathbf{p_j}$ is a point of (respectively, in the

interior of) R_i, then $S(\mathbf{p_i}) \leq S(\mathbf{p_j})$ ($S(\mathbf{p_i}) < S(\mathbf{p_j})$). Consequently, if there are at least k points of C in the hyper-rectangle of o_i, then no strictly monotone scoring function can make o_i one of the k highest scored objects.

Now, let us say that object o_i is a *potential k best match* iff there are exactly $k-1$ points in its hyper-rectangle, and simply a potential best match if it is a potential 1 best match (i.e., no points are found in R_i). Consider now a top k query $\mathcal{Q}$ using a scoring function S. Although it is plain to see, due to the monotonicity of S, that each object in the result of $\mathcal{Q}$ will be a potential j best match, with $j \leq k$, it is also true that $\mathcal{Q}$ will likely miss many potential best matches, in particular those in the parts of the answer space where S does not attain high values.[2] This lack of an "overall view" of the potential best matches is usually handled by means of relevance feedback techniques, that iteratively modify the parameters (i.e., weights) of the scoring function so as to better fit user preferences and lead the search to focus on the more relevant parts of the answer space [20]. However, relevance feedback mechanisms require a good "starting point" to work well[3] and several refinement steps are usually needed before weights converge to the "right" values, a fact which can severely degrade system performance [5].[4] Further, commonly used scoring function, such as weighted average, are necessarily prone to miss some potential best match, *no matter how their weights are adjusted*. To see why this is the case, take as scoring function a weighted average:

$$s_i = \text{avg}_W(\mathbf{p_i}) = \text{avg}_q\{w_q \times s_{i,q}\} \tag{1}$$

where W is a weight vector, and assume $k = 1$ and 2 sub-queries, i.e., $m = 2$ (see also Figure 1 (b)). Let object o_0 be a potential best match, with scores $\mathbf{p_0} = (s_{0,1}, s_{0,2})$, where both $s_{0,1}$ and $s_{0,2}$ are less than 1, and assume that collection C also includes 2 objects o_1 and o_2 such that $\mathbf{p_1} = (s_{0,1} + \delta, s_{0,2} - \epsilon)$ and $\mathbf{p_2} = (s_{0,1} - \epsilon, s_{0,2} + \delta)$, with $\delta > \epsilon > 0$. It is not difficult to see that *any* choice of the weight values in Equation (1) will either yield $\text{avg}_W(\mathbf{p_0}) < \text{avg}_W(\mathbf{p_1})$ or $\text{avg}_W(\mathbf{p_0}) < \text{avg}_W(\mathbf{p_2})$, thus always inhibiting object o_0 to be retrieved.

[2] As a simple example, if o_i is a potential best match that has a single (very) low partial score and the scoring function is min, then the overall score of o_i will be (very) low as well.

[3] Clearly, this depends not only on the scoring function itself but also on the actual content of the database, which might lead to a result set where the user cannot find any relevant object at all.

[4] Note that this is orthogonal to the problem of efficiently answering a single query. Even when queries can be efficiently evaluated, answering many of them will anyway become a performance bottleneck.

This negative result, that can easily be extended to arbitrary k and m values as well as to many other scoring functions [3], coupled with the observations on the inherent complexity paid for effectively exploring the space of potential best matches, suggests us to look for a more general and efficient alternative to the integration problem.

3. Qualitative Preferences

As it is well-known from decision theory, preferences do not always admit a numerical representation through a scoring function [17]. In order to rank objects it is indeed sufficient that preferences are "qualitatively" defined through a *preference relation*, precisely defined as follows.

Definition 2 (Preference Relation) *Let X be a domain of values. A* preference relation *over X is a binary relation $\succ \subseteq X \times X$. If $x_1, x_2 \in X$ and $(x_1, x_2) \in \succ$, we also write $x_1 \succ x_2$ and say that x_1 is preferable to x_2 or, equivalently, that x_1* dominates *x_2. If neither $x_1 \succ x_2$ nor $x_2 \succ x_1$ hold, we say that x_1 and x_2 are* incomparable*, written $x_1 \sim x_2$.*

Coherently with the model introduced in Section 2, preferences are defined over the answer space (thus $A \equiv X$ in above definition); however, slightly abusing notation, we will also write $o_i \succ o_j$ whenever $\mathbf{p_i} \succ \mathbf{p_j}$ holds.

Although it is immediate to see that preference relations include scoring functions as a special case, since for any S one can immediately define a corresponding $\succ_S$ as $o_i \succ_S o_j \Leftrightarrow S(\mathbf{p_i}) > S(\mathbf{p_j})$, Definition 2 is far too general for our purposes. Indeed, it is still reasonable to demand that qualitative preferences, even if not based on the comparison of overall scores, still have some kind of monotonic behavior with respect to partial scores, so as to ensure that "doing better" on sub-queries will not worsen the overall goodness of an object.

Definition 3 (Monotonicity of preference relations) *A preference relation $\succ$ over the answer space $A = [0,1]^m$ is* monotone *if $s_{j,q} \leq s_{i,q}$ $\forall q$ implies $\mathbf{p_j} \not\succ \mathbf{p_i}$, and* strictly monotone *if $s_{j,q} < s_{i,q}$ $\forall q$ implies $\mathbf{p_i} \succ \mathbf{p_j}$.*

Finally, we also want to preserve the *transitivity* of $\succ$, that is, $x_1 \succ x_2$ and $x_2 \succ x_3$ imply $x_1 \succ x_3$. This, together with monotonicity, ensures that $\succ$ is also a *strict partial order* (PO for short), thus transitive (by hypothesis) and *irreflexive*, that is, $x \not\succ x$ (by Definition 3).[5]

[5] Note that a strict partial order is not necessarily monotone with respect to partial scores and, going the other way, monotonicity is not enough to guarantee

As to the difference between monotone and strictly monotone preference relations, we observe that the first ones have the pitfall of being potentially insensitive to changes in partial scores, that is, being o_i better than o_j on *all* sub-queries would not guarantee $o_i \succ o_j$. We find this very counter-intuitive, the reason of why in the sequel we will only consider strictly monotone PO preference relations.

The first example of a strictly monotone PO preference relation are the so-called *Skyline preferences* [8].

Definition 4 (Skyline preferences) *The Skyline preference relation* $\succ_{SL}$ *over* $A = [0,1]^m$ *is defined as:*

$$o_i \succ_{SL} o_j \Leftrightarrow (\forall q : s_{j,q} \leq s_{i,q}) \wedge (\exists q : s_{j,q} < s_{i,q}) \tag{2}$$

Thus, o_i is preferred to o_j iff it is at least as good as o_j on all sub-queries and there is at least one sub-query for which o_i performs better than o_j. The set of objects of C for which there is no object that dominates them according to $\succ_{SL}$ is called the *Skyline* of C.

Skyline preferences coincide with the notion of *Pareto optimality* from decision theory, where the Skyline is known as the Pareto set. Its importance is that, if an object is a *potential best match* (as defined in Section 2.1) then it will be an element of the Pareto set and, conversely, each Pareto-optimal point is a potential best match. Intuitively, the Skyline provides us with an "overall view" of the potential best objects for a given query, a fact that highly simplifies the task of focusing on the part of the search space containing more relevant objects.

Figure 2 shows the results of a sample query over an image database when Skyline preferences are used to integrate the results of two sub-queries, over color and texture features, respectively. The figure shows the target "eagle" image $\mathcal{Q}$ and its potential best matches. Note that most of them belong to the same semantic class, "Birds", of $\mathcal{Q}$ (more details on this point are given in Section 5) and that they are quite spread over the answer space.

Clearly, Skylines are not the whole story about qualitative preferences. Without distracting the reader with too many details on the engineering of complex preferences (see, e.g., [18]), in the following we concentrate on a kind of preferences based on the concept of *priority* among regions of the answer space.

To start with, consider the case where o_{bad} is an object with scores, say, $\mathbf{p_{bad}} = (0.8, 0.01, \ldots, 0.01)$, and assume that 0.8 is the best score for sub-query Q_1, with no other object obtaining such score. Regardless

---

the transitivity of a preference relation (the same is true for strict monotonicity). This is why both hypotheses need to be independently stated.

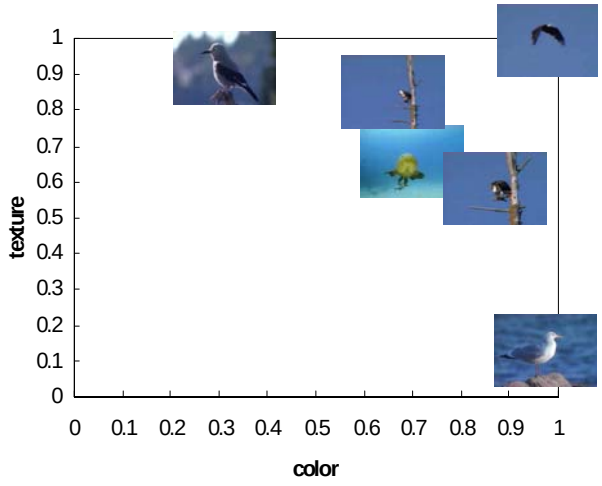


Figure 2. The Skyline of the "eagle" image

of the poor partial scores $s_{bad,q} = 0.01$ $(q = 2, \ldots, m)$, this alone is sufficient to guarantee, according to Definition 4, that o_{bad} will be part of the Skyline of $\mathcal{Q}$. This could be questionable to the user, especially if the database contains (possibly many) other objects with somewhat "more balanced" score values. On the other hand, the same user could be interested in seeing o_{bad} if no such alternative solutions are currently available. Imposing hard constraint on partial scores clearly does not work here, since the problem is to return to the user the "best" results compatible with the *actual* contents of the database. But this is exactly the behavior that can be easily achieved working with qualitative preferences!

Let $Y = \{A_1, \ldots, A_P\}$ be a partition of the answer space and define a strictly monotone PO preference relation, $\succ_Y$, among regions of Y.

Definition 5 (RS preferences) *Let $Reg : A \rightarrow Y$ be a function that maps each point of A into its (unique) region of Y. The Region-prioritized Skyline (RS) preference relation $\succ_{RS}$ over $A = [0,1]^m$ is defined as:*

$$o_i \succ_{RS} o_j \;\Leftrightarrow\; (Reg(\mathbf{p_i}) \succ_Y Reg(\mathbf{p_j})) \vee ((Reg(\mathbf{p_i}) = Reg(\mathbf{p_j})) \wedge (\mathbf{p_i} \succ_{SL} \mathbf{p_j}))$$

Thus, within a same region the Skyline logic applies, whereas priority among regions prevails if two points belong to different regions. As a simple example (see also Figure 3 (a)), let $m = 2$ and $Y = \{A_1, A_2\}$, with $A_1 = [0,1] \times [0.7,1]$, $A_2 = [0,1] \times [0,0.7)$, and $A_1 \succ_Y A_2$. Any point in the "upper rectangle" A_1 will dominate points in the "lower

rectangle" A_2. Intuitively, this will favor objects with a good partial score for sub-query Q_2. Among such objects (if any), the best matches will be determined using Skyline preferences. If region A_1 is empty, then the best matches will be found in region A_2.

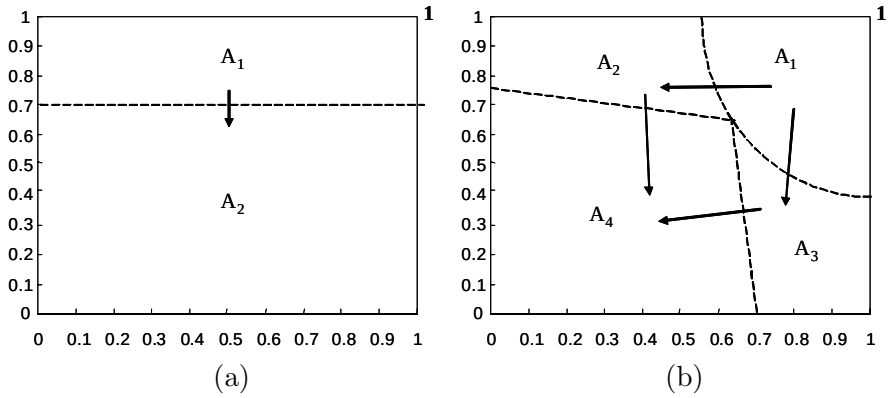


Figure 3. (a) A simple example of RS preferences, where the arrow means that region A_1 is preferred to region A_2; (b) An illustration of Lemma 1, $Y = \{A_1, A_2, A_3, A_4\}$

In order to guarantee the strict monotonicity of $\succ_{RS}$ we need to ensure that priority among regions does not contrast with Skyline preferences.

Lemma 1 *The preference relation $\succ_{RS}$ is strictly monotone iff whenever $Reg(\mathbf{p_i}) \neq Reg(\mathbf{p_j})$ and $s_{j,q} < s_{i,q}$ $\forall q$, then it is $Reg(\mathbf{p_i}) \succ_Y Reg(\mathbf{p_j})$.*

Proof.
(only if) Immediate from Definitions 3 and 5.
(if) We consider two cases: (a) if $Reg(\mathbf{p_i}) = Reg(\mathbf{p_j})$ then the result follows from the strict monotonicity of $\succ_{SL}$; (b) if $Reg(\mathbf{p_i}) \neq Reg(\mathbf{p_j})$ and $s_{j,q} < s_{i,q}$ for all q, by hypothesis $Reg(\mathbf{p_i}) \succ_Y Reg(\mathbf{p_j})$, thus $o_i \succ_{RS} o_j$, as required. □

Figure 3 (b) provides a graphical intuition on above Lemma, where arrows denote preferences among regions. Note also that $A_1 \succ_Y A_4$ follows by transitivity.

Although, for the sake of definiteness, we have combined region prioritization with Skyline preferences, in Equation 3 one could use within each region *any* strictly monotone PO preference relation to compare objects, and still obtain a valid strictly monotone PO region-prioritized preference relation. For instance, one could define $Y = \{A_1, A_2, A_3, A_4\}$, and within each region use a, possibly different (!), preference relation, say $\succ_{SL}$ in A_1, $\succ_{\min}$ in A_2, and so on.

4. Query Evaluation

Efficient evaluation of complex multimedia queries is a challenging task which requires to combine advanced indexing techniques, for picking the best objects for each sub-query, with smart integration algorithms, in order to avoid unnecessarily picking too many objects to correctly determine the final result. Although methods exist that try to process complex queries as a whole, i.e., without splitting them into sub-queries and then using a single centralized index, they have a limited applicability and/or are bound to specific access methods for the evaluation of sub-queries. For instance, the method in [12] applies only to multi-object queries on a given metric space, thus it is unsuitable if one specifies on-the-fly which are the objects' features to be used for querying purposes. On the other hand, the method described in [7], that can easily support queries on arbitrary subsets of the objects' features, only works if all sub-queries are on a vector space and objects are indexed by a VA-file, thus making restrictive hypotheses on objects' representation and on the underlying structures used for sub-queries evaluation.

In order to stay as general as possible, we base our approach on a well-accepted access model that, since the seminal paper by Fagin [14], has become a standard for the design of integration algorithms (see, e.g., [16, 15, 2, 1]). According to such model, evaluating each sub-query Q_q ($q = 1, \ldots, m$) yields a *ranked list* L_q of pairs $(o_i, s_{i,q})$, containing all objects in the collection C ordered by descending score values. Relevant information can be retrieved through one of two distinct modalities: A *sorted access* retrieves from a list L_q the next unseen object on that list, say o_i, together with its partial score, $s_{i,q}$; a *random access*, on the other hand, given an object o_i seen via sorted access on some list L_q, retrieves from the database the needed features and, consequently, evaluates the missing partial scores for o_i. The *cost* of an integration algorithm is then taken to be the (possibly weighted) sum of the number of sorted and random accesses performed by the algorithm for delivering the query result set.

4.1. The MPO Algorithm

Our first algorithm, called MPO and described in Figure 4, is based on the semantics of the Best operator [22] that, given an input collection C and a preference relation $\succ$, returns all the potential best matches in C:

$$\beta_{\succ}(C) = \{o \in C \mid \nexists o' \in C, o' \succ o\} \tag{3}$$

The logic of algorithm MPO can be explained as follows. At each step MPO retrieves via sorted access (step 4) the best "unseen" object

Algorithm MPO (Input: query $\mathcal{Q}$, collection C, preference relation $\succ$)

(1) Set $Result = \emptyset$; Set $\underline{\mathbf{p}} = (1, \ldots, 1)$; /* $\underline{\mathbf{p}}$ is the threshold point */
(2) While $(\nexists (o_i, \mathbf{p_i}) \in Result$ such that $\mathbf{p_i} \succ \underline{\mathbf{p}})$:
(3) For each sub-query Q_q $(q = 1, \ldots, m)$ do:
(4) Retrieve the next unseen object o_i from L_q; /* sorted access */
(5) Retrieve missing scores for the other sub-queries and obtain $\mathbf{p_i}$; /* random accesses */
(6) Set $Dominated$ = false;
(7) While (not($Dominated$) $\wedge\ \exists\ (o_j, \mathbf{p_j}) \in Result$ unmatched with $\mathbf{p_i}$):
(8) Compare $\mathbf{p_i}$ with $\mathbf{p_j}$: $\begin{cases} \mathbf{p_i} \succ \mathbf{p_j} & \text{remove } (o_j, \mathbf{p_j}) \text{ from } Result, \\ \mathbf{p_i} \sim \mathbf{p_j} & \text{do nothing,} \\ \mathbf{p_j} \succ \mathbf{p_i} & \text{set } Dominated = \text{true;} \end{cases}$
(9) End While;
(10) If not($Dominated$) insert $(o_i, \mathbf{p_i})$ in $Result$;
(11) Let $\underline{s_q}$ be the lowest score seen by sorted access on list L_q; Set $\underline{\mathbf{p}} = (\underline{s_1}, \ldots, \underline{s_m})$;
(12) End For;
(13) End While;
(14) Return $Result$.

Figure 4. The MPO algorithm

o_i from one of the m sorted lists, and then obtains missing partial scores for such object via random access (step 5). The so-obtained representative point $\mathbf{p_i}$ is then compared with the current objects in $\beta_\succ(C)$ (steps 7 and 8). If no objects o_j dominates o_i, o_i is inserted in $\beta_\succ(C)$ (possibly also removing objects dominated by o_i itself), otherwise o_i is discarded. At each point MPO maintains a "threshold point" $\underline{\mathbf{p}}$, whose q-th component, $\underline{s_q}$, is the lowest partial score seen so far under sorted access on list L_q. As soon as an object o_i is found such that $\mathbf{p_i}$ dominates the threshold point $\underline{\mathbf{p}}$ the algorithm terminates.

Theorem 1 *The MPO algorithm correctly computes $\beta_\succ(C)$ for any strictly monotone PO preference relation $\succ$ that is* Pareto-consistent*, that is, $\succ$ satisfies the implication:*
$(x_1 \succ x_2) \wedge (x_2 \succ_{SL} x_3) \Rightarrow (x_1 \succ x_3)$.

Proof.
($\beta_\succ(C) \subseteq Result$). If an object $o_j \in \beta_\succ(C)$ has been retrieved via sorted access before the algorithm stops, then it is guaranteed that it will also belong to the final result. Thus, assume by contradiction $o_j \in \beta_\succ(C)$, yet o_j has not been seen by the algorithm. Let o_i be the object that is found at step 2 to dominate the threshold point. Unless $\mathbf{p_j}$ is coincident with the threshold point, in which case we are obviously done, at least one partial score of o_j is strictly less than the corresponding threshold value. It follows that $\underline{\mathbf{p}} \succ_{SL} \mathbf{p_j}$. Since $\succ$ is

assumed to be Pareto-consistent, we also have $\mathbf{p_i} \succ \mathbf{p_j}$ (since $\mathbf{p_i} \succ \underline{\mathbf{p}}$), thus $o_j \notin \beta_{\succ}(C)$.
($Result \subseteq \beta_{\succ}(C)$). We prove that if $o_j \notin \beta_{\succ}(C)$ then $o_j \notin Result$. If o_j has not been seen by the algorithm then it cannot be part of the final result. Thus, assume o_j has been seen. Since $o_j \notin \beta_{\succ}(C)$ and $\succ$ is a strict partial order there is at least one object $o_i \in \beta_{\succ}(C)$ such that $o_i \succ o_j$. Since we have already proved that $\beta_{\succ}(C) \subseteq Result$, such o_i has been seen, and steps 7 and 8 of the algorithm guarantee that o_i and o_j have been compared. Therefore, o_j cannot belong to $Result$. □

The hypothesis of "Pareto-consistency" has a really negligible impact on the general applicability of the MPO algorithm. Indeed, any "reasonable" strictly monotone PO preference relation is also Pareto-consistent. For instance, this is easily shown to be the case if $\succ$ represents a strictly monotone scoring function S (i.e., $\succ \equiv \succ_S$). Also, Skyline (SL) and Region-prioritized Skyline (RS) preferences are both Pareto-consistent (for SL the result is obvious and for RS can also be easily derived). A preference relation that is not Pareto-consistent would have indeed a rather strange behavior: A point $\mathbf{p_i}$ is preferred to point $\mathbf{p_k}$ but is not preferred to another point $\mathbf{p_j}$ whose partial scores are all less than or equal to those of $\mathbf{p_k}$ (with at least one strict inequality)![6]

4.2. The Incremental MPO Algorithm

If one wants to explicitly control the cardinality of the result of a query the Best operator (thus, algorithm MPO) is not the right choice, since it returns *all and only* the potential best matches, regardless of how many they are. For instance, it is known that the size of the Skyline can become quite large, and grows fast with the number of dimensions when partial scores have a negative correlation [8]. On the other hand, when the size of $\beta_{\succ}(C)$ becomes too small it would be advisable to allow the user to retrieve also further "good" objects, even if they are not in $\beta_{\succ}(C)$.

To achieve both goals we start by introducing a new operator, called BesTop, that combines the semantics of Best and Top-k operators. For its definition it is first useful to remind the "layered" version of the

[6] However, extending algorithm MPO so as to process also non Pareto-consistent preferences is easy: Just perform, after $\mathbf{p_i} \succ \underline{\mathbf{p}}$ has been verified, some further sorted accesses on all lists, so as to see on all of them a partial score strictly less than that of the threshold point (which now stays fixed). At this point, since $\succ$ is strictly monotone, all unseen objects $\mathbf{p_j}$ are guaranteed to be dominated by $\underline{\mathbf{p}}$.

Best operator [22]:[7]

$$\beta^1_{\succ}(C) = \beta_{\succ}(C) \quad (4)$$

$$\beta^{\ell+1}_{\succ}(C) = \beta_{\succ}(C - \cup_{i=1}^{\ell}\beta^i_{\succ}(C)) \quad (5)$$

Thus, $\beta^{\ell}_{\succ}(C)$ retrieves the ℓ-th "layer" of (the partial order induced by $\succ$ on) C. Figure 5 provides an intuition of "what is below" the Skyline of the "eagle" query image shown in Figure 2, with dotted lines separating the layers of the partial order induced by SL preferences.

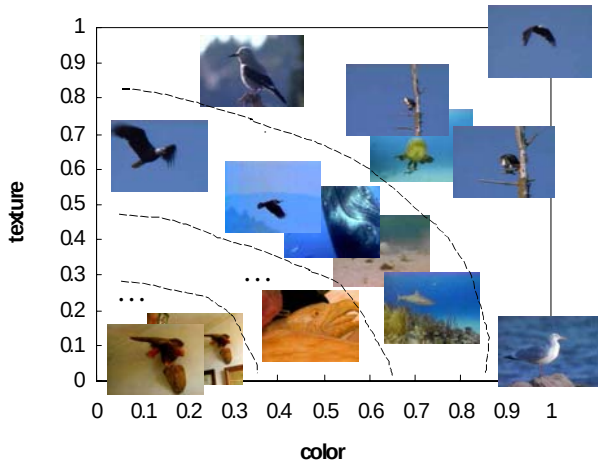


Figure 5. Some layers of the partial order induced by SL preferences for the "eagle" query image

Definition 6 (BesTop operator) *Let $\ell(k) \geq 1$ be the smallest integer that satisfies the inequality $\sum_{i=1}^{\ell(k)} \mid \beta^i_{\succ}(C) \mid \geq k$. The BesTop operator $\beta^{[k]}_{\succ}(C)$ retrieves k objects from C such that:*

- *$\beta^{[k]}_{\succ}(C)$ includes all the objects in the first $\ell(k) - 1$ layers of C;*
- *it further includes other $k - \sum_{i=1}^{\ell(k)-1} \mid \beta^i_{\succ}(C) \mid$ objects from the $\ell(k)$-th layer of C.*

A naïve approach to compute $\beta^{[k]}_{\succ}(C)$ would be to extend algorithm MPO so that it can return all the first $\ell(k)$ layers (rather than only the first one), and then to select all the objects in the first $\ell(k) - 1$ layers plus others from layer $\ell(k)$, so as to reach the desired result cardinality

[7] An analogous extension has been proposed for the Winnow operator [11].

k.[8] The major drawback of this approach is made evident through a simple example. Let $k = 1$, thus $\ell(1) = 1$ (the first layer is obviously enough). If we run MPO and wait until its completion we would miss the opportunity to stop as soon as we can conclude that *a single object* belongs to $\beta^1_{\succ}(C)$. In general, we see that no object of a layer ℓ can be returned by MPO before it is discovered that no further objects belong to layer ℓ, which might severely affect performance.

Algorithm iMPO (incremental MPO), summarized in Figure 6, elegantly solves the above problem. To understand the logic of iMPO the following observation is useful.

Observation 1 *Let $\succ$ be a strictly monotone PO preference relation that is Pareto-consistent. If $\underline{\mathbf{p}} \not\succ \mathbf{p_i}$ and $\underline{\mathbf{p}} \succ_{SL} \mathbf{p_j}$ then $\mathbf{p_j} \not\succ \mathbf{p_i}$.*

Indeed, if $\mathbf{p_j} \succ \mathbf{p_i}$ then from the hypothesis of Pareto consistency it would follow that $\underline{\mathbf{p}} \succ \mathbf{p_i}$, thus contradicting the hypothesis.

Algorithm iMPO (Input: query $\mathcal{Q}$, collection C, pref. relation $\succ$, integer k)

(1) Set $NoOfResults = 0$; Set $ThisLayer = NextLayer = \emptyset$; Set $\underline{\mathbf{p}} = (1, \ldots, 1)$;
(2) While $(NoOfResults < k)$:
(3) While $(\nexists (o_i, \mathbf{p_i}) \in ThisLayer$ such that $\mathbf{p_i} \succ \underline{\mathbf{p}} \wedge NoOfResults < k)$:
(4) For each sub-query Q_q $(q = 1, \ldots, m)$ do:
(5) Retrieve the next unseen object o_i from L_q;
(6) Retrieve missing scores for the other sub-queries and obtain $\mathbf{p_i}$;
(7) Set $Dominated$ = false;
(8) While (not($Dominated$) $\wedge\ \exists\ (o_j, \mathbf{p_j}) \in ThisLayer$ unmatched with $\mathbf{p_i}$):
(9) Compare $\mathbf{p_i}$ with $\mathbf{p_j}$: $\begin{cases} \mathbf{p_i} \succ \mathbf{p_j} & \text{move } (o_j, \mathbf{p_j}) \text{ from } ThisLayer \text{ to } NextLayer, \\ \mathbf{p_i} \sim \mathbf{p_j} & \text{do nothing,} \\ \mathbf{p_j} \succ \mathbf{p_i} & \text{set } Dominated = \text{true; insert } (o_i, \mathbf{p_i}) \text{ in } NextLayer; \end{cases}$
(10) End While;
(11) If not($Dominated$) insert $(o_i, \mathbf{p_i})$ in $ThisLayer$;
(12) Let $\underline{s_q}$ be the lowest score seen by sorted access on list L_q; Set $\underline{\mathbf{p}} = (\underline{s_1}, \ldots, \underline{s_m})$;
(13) Output all objects $(o_i, \mathbf{p_i}) \in ThisLayer$ s.t. $\underline{\mathbf{p}} \not\succ \mathbf{p_i}$ and update $NoOfResults$;
(14) End For;
(15) End While;
(16) If $(NoOfResults < k)$ then: /* starts to process the next layer */
(17) Set $ThisLayer = NextLayer$; Set $NextLayer = \emptyset$;
(18) For all $o_i, o_j \in ThisLayer$ s.t. $\mathbf{p_i} \succ \mathbf{p_j}$: move $(o_j, \mathbf{p_j})$ from $ThisLayer$ to $NextLayer$;
(19) Output all objects $(o_i, \mathbf{p_i}) \in ThisLayer$ s.t. $\underline{\mathbf{p}} \not\succ \mathbf{p_i}$ and update $NoOfResults$;
(20) End If;
(21) End While.

Figure 6. The incremental MPO algorithm

iMPO exploits the above observation as follows. Each time the threshold point $\underline{\mathbf{p}}$ changes, iMPO checks if some object o_i that has already

[8] Note that at layer $\ell(k)$ ties are arbitrarily broken, as it is customary for top k queries.

been retrieved is *not* dominated by $\underline{\mathbf{p}}$ (step 13), in which case o_i can be immediately returned to the user. For this reason we call $\underline{\mathbf{p}} \not\succ \mathbf{p_i}$ the *delivery condition* of iMPO for object o_i.

The second major feature that distinguishes iMPO from MPO is the management of multiple layers of the partially ordered collection C. Rather than simply removing objects that are found to be dominated by some other object (as MPO does), iMPO keeps them in a $NextLayer$ structure. Objects in such a structure are processed again upon completion of a layer, and before restarting to retrieve other objects via sorted access (steps 17 and 18).

It is a fact that iMPO will always return an object o_i before MPO does so. This stems from the following basic result, in which we consider MPO extended so as to deal with multiple layers (rather than just the first one) of C. Having already introduced iMPO, this extension is straightforward, since it essentially amounts to dropping the incremental delivery condition from the logic of iMPO.

Lemma 2 *Let o_j be an object at layer ℓ of C, and let o_i be the object in the same layer that allows MPO to terminate the elaboration of the ℓ-th layer (thus, $\mathbf{p_i} \succ \underline{\mathbf{p}}$). Let $\#SA(MPO)$ be the number of sorted accesses that MPO has executed up to this point, and $\#SA(iMPO)$ those executed by iMPO when it delivers object o_j. It is $\#SA(iMPO) \leq \#SA(MPO)$.*

Proof.
We have $\mathbf{p_i} \succ \underline{\mathbf{p}}$ and, from the hypothesis that o_i and $_j$ belong to the same layer, $\mathbf{p_i} \sim \mathbf{p_j}$. From these we can conclude that $\underline{\mathbf{p}} \not\succ \mathbf{p_j}$, which proves the result. □

5. Experimental Analysis

In this section we first quantify the advantages obtained by using algorithm iMPO in place of MPO, after that we compare qualitative preferences (namely, Skyline (SL) and Region-prioritized Skyline (RS) preferences) with scoring functions (namely, min and avg) in terms of both effectiveness and efficiency. Finally, we compare qualitative preferences with the *median rank* criterion, which is an original integration rule recently proposed in [15].

The results we present are obtained using a real-world image collection consisting of about 10,000 color images. Although this data set is not particularly large, we chose it for two reasons: Since each image comes with a manually assigned semantic classification into one

of 7 classes, this allows us to evaluate effectiveness (quality) of results, which would not be possible without an objective "ground truth". To this end, given a query image, any image in the same class of the query is considered relevant, whereas all other images are considered not relevant, regardless of their actual low-level feature contents. Note that classes are just used for evaluation purposes and not during the retrieval phase (i.e., algorithms know nothing about the class of an image). This leads to hard-to-solve conceptual queries, since within a same class feature values may wildly vary. Further, since in this paper we are not dealing with issues related to the evaluation of sub-queries, the actual size of the data set is not particularly relevant in assessing performance. Indeed, although our system uses indexes to efficiently evaluate sub-queries, relative figures are not shown here (any method able to return ranked lists would serve the purpose).

We implemented algorithms MPO (extended so as to answer "first ℓ layers" queries) and iMPO algorithms in C++ on top of Windsurf [4]. Windsurf is an advanced region-based image retrieval system that, using wavelet transform, automatically segments each image into a set of homogeneous *regions*, based on the proximity of wavelet coefficients, which convey information about color and texture features. Each region corresponds to a cluster of pixels and is represented through a 37-dimensional feature vector.[9] On average, 4 regions were obtained from each image in our collection. The same procedure is adopted when an image query $\mathcal{Q}$ is submitted. If m is the number of regions extracted from $\mathcal{Q}$, each of the m regions becomes a sub-query. Partial scores for a given query region Q_q are obtained by using a distance function based on the *Bhattacharyya metric* [6], which is commonly used to compare ellipsoids.

All the results we present are averaged over a sample of 100 randomly-chosen query images. The specific metrics we use to evaluate performance of algorithms and/or preferences are as follows.

Efficiency. We measure the number of sorted accesses, $\#SA$, and the number of random accesses, $\#RA$, executed to return the result of a query. This ensures a fair, system-independent, comparison. Note that actual execution times are indeed expected to vary in a significant way depending on the relative cost of sorted and random accesses, the nature of the underlying system(s) evaluating sub-queries (e.g., Web-based or not), the available access methods, etc. To avoid distracting the reader with too many parameters and variables which would

[9] In detail: 12 dimensions are used for the cluster centroid (3 color channels $\times$ 4 frequency sub-bands), 24 coefficients store the 3×3 (symmetric) covariance matrices of the 4 sub-bands, and 1 coefficient represents the cluster size.

consequently come into play we opted for clean, easy to understand, metrics.

Effectiveness. As possible measures of how good the results of using a specific integration rule are, we consider the classical *precision* metric (i.e., the percentage of relevant images found by a query) and the extent to which relevant images are representative of the query class, that is, how well they fit the actual distribution of all images in the query class. This allows a finer assessment of the quality of results that precision alone cannot provide.

In order to generate RS preferences we proceed as follows (see also Definition 5 in Section 3). On each of the m coordinates of the answer space A, we set a "soft threshold" θ_q $(0 < \theta_q < 1)$ and assign a 0 bit to the "below-threshold" interval $[0, \theta_q)$ and a 1 bit to the "above-threshold" interval $[\theta_q, 1]$. This leads to a partition Y of 2^m regions, each univocally represented by an m-bit binary code. Given regions A_i and A_j, the preference relation for such regions is

$$A_i \succ_Y A_j \Leftrightarrow code(A_i) \wedge code(A_j) = code(A_j)$$

where bitwise AND is used and $code(A_i)$ is the binary code of region A_i. For instance, when $m = 4$, this says that the region with code 1011 dominates the region with code 1000, whereas it is indifferent to region 0100. Since $\succ_Y$ defines a Boolean lattice over regions (with region $11\ldots1$ being the best region and $00\ldots0$ the worst one) it is easy to show that Lemma 1 is satisfied, thus $\succ_{RS}$ is a strictly monotone PO.

Although we experimented with several combinations of soft threshold values, here we just report results for the case $\theta_q = 0.4$ $\forall q$.

5.1. Experimental Results

Experiment 1: The aim of our first experiment is to measure the relative efficiency of iMPO versus MPO, when both algorithms are using a same preference relation. This guarantees that MPO and iMPO return the same set of objects, although at different times.

Our results confirm that iMPO consistently outperforms MPO. In Figure 7 we show the number of sorted accesses executed to answers a specific query. Results for other queries and for the number of random accesses follow a similar behavior. MPO, by its nature, delivers objects in bursts, each burst corresponding to the termination of one layer. For instance, in Figure 7 (a) MPO needs 628 sorted accesses to return all the 27 images in the 1st layer of the Skyline, 956 to complete the 2nd layer, and so on. A somewhat bursty behavior is also observed with iMPO, starting from the 2nd layer. To explain this, consider that even if *all* objects in the current layer have been output, iMPO still needs

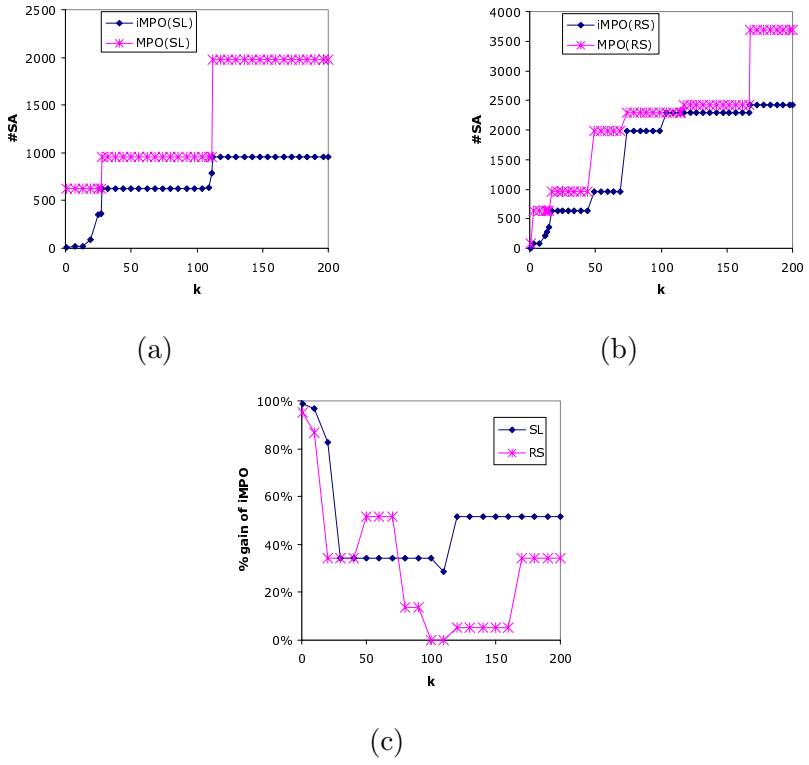
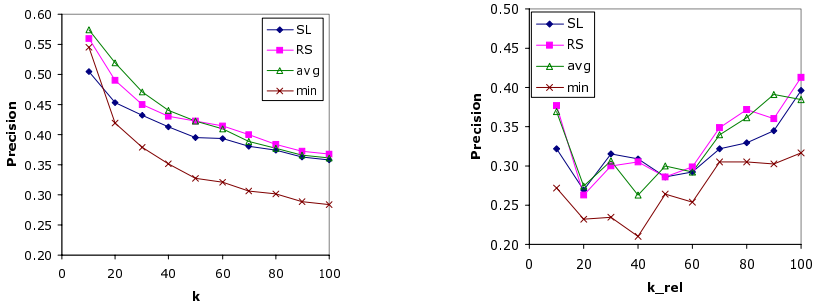


Figure 7. Sorted accesses of MPO and iMPO algorithms for a specific query: (a) Skyline (SL) preferences; (b) Region-prioritized Skyline (RS) preferences. In (c) the percentage gain of iMPO over MPO is shown. The abscissa reports the no. of retrieved objects k

to wait that the test $\mathbf{p_i} \succ \underline{\mathbf{p}}$ succeeds before moving to the next layer. This "waiting time" leads to accumulate objects in the *NextLayer*, most of which are subsequently delivered as soon as the test succeeds. Nonetheless, Figure 7 (c) shows that the gain in efficiency of iMPO over MPO is remarkable; therefore in the sequel we do not consider MPO anymore and always use iMPO.

Experiment 2: In this second series of experiments our goal is to compare qualitative preferences (i.e., SL and RS) and scoring functions (i.e., min and avg) in terms of quality of results. Figure 8 (a) shows precision values versus the number of retrieved object k. It can be seen that SL, and RS in particular, preferences attain precision levels comparable to that of avg, whereas min has a definitely poor behavior. Figure 8 (b) shows similar results for different *recall* values, as measured by the number of *relevant* objects retrieved, k_{rel}.



(a) (b)

Figure 8. Precision vs no. of retrieved objects (a) and no. of relevant retrieved objects (b)

Clearly, similar precision values do not imply similar results. Indeed, the sets of relevant objects retrieved by means of qualitative preferences and scoring functions share, on the average and for any value of k, less than 50% common elements. This observation motivates a more detailed investigation on which is the difference, in terms of relevant objects retrieved, of using qualitative preferences in place of scoring functions since, at least in principle, SL and RS preferences should be able to provide a better "overall view" of the objects relevant to a query. In order to quantify this concept, we consider the *distribution of relevant objects over the answer space*. By comparing the distributions of the relevant objects returned by the different integration rules, it is possible to establish which one better fits the distribution of all relevant images in the query class (thus, which one better represents the actual class contents).

To properly compare the distributions of relevant results, we use an information-theoretic measure, related to the cross-entropy of two distributions, known as the *Kullback-Leibler (KL) divergence* [13]. Given a reference distribution f and a test one g, the KL divergence of g with respect to f is defined as

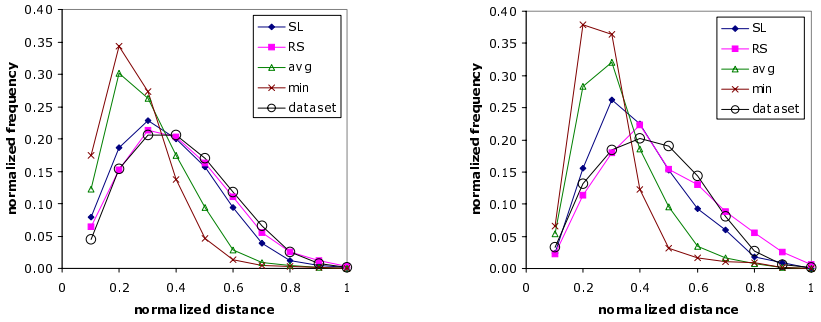
$$KL(g; f) = \int_x f(x) ln \left(\frac{f(x)}{g(x)} \right) dx.$$

Note that $KL(g; f) \geq 0$, with 0 attained only if $g = f$. Thus, $KL(g_1; f) < KL(g_2; f)$ denotes that g_1 fits f better than g_2.

In our case, we take f to be the distance distribution of *all* the relevant objects for a query $\mathcal{Q}$, and the g_i's be the (approximate) distance distributions of the relevant objects returned in the first $k = 100$ results. All distances are measured over the answer space, by computing

the Euclidean distance between the representative points of relevant images.

Figure 9 (a) shows the (averaged over all query images) actual distribution of the whole data set (label dataset in the figure) and those of min, avg, SL, and RS. Figure 9 (b) does the same but just for queries of a specific class ("TreeLeaves").



(a) (b)

Figure 9. Global (a) and relative to the "TreeLeaves" class (b) distance distributions of relevant objects

Table II synthesizes everything using KL divergence values.

Table II. Kullback-Leibler (KL) divergence values for the distributions in Figure 9

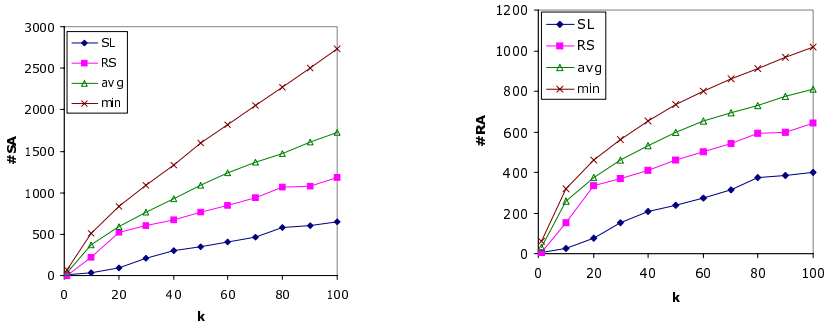
	SL	RS	avg	min
Global	0.032	0.006	0.297	0.550
TreeLeaves	0.050	0.031	0.312	0.680

It is evident that RS preferences, besides leading to precision values comparable to those of avg, have a remarkably better capability to reflect the actual distribution of relevant objects. Skyline preferences are, to this end, slightly worse, even if divergence values are still one order of magnitude better than those of avg and min.

An important advantage derived from having a small value of KL is related to the implementation of effective relevance feedback mechanisms. Since all these methods share the idea of exploiting the user feedback (given on the query outcome) in order to refine the initial query, giving to the user a more accurate "overall view" of the content of the query class it makes possible to cut down the number of user-

system interactions needed to lead to acceptable results. We plan to thoroughly investigate this issue in the prosecution of our research.

Experiment 3: In the third experiment our objective is to analyze the efficiency of iMPO in answering top k queries. Figure 10 shows how many sorted and random accesses are needed by iMPO to deliver k objects, depending on the specific preferences used. In this case SL is undoubtedly the winner, saving up to about 70% and 80% database accesses against avg and min, respectively. Efficiency of RS is slightly poorer, however reaching a performance level that is always better than that of both avg and min (35% and 60% speed-up, respectively).



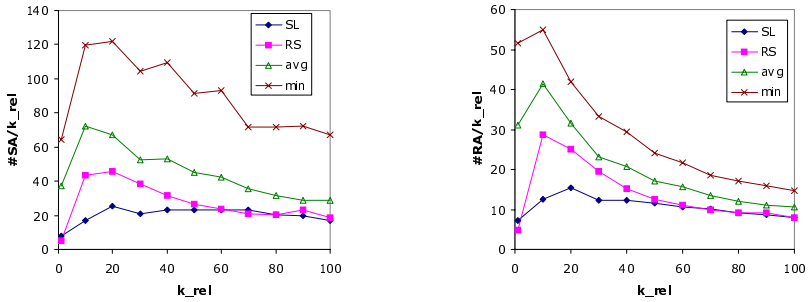
(a) (b)

Figure 10. Sorted accesses (a) and random accesses (b) vs no. of retrieved objects (k)

The reason why SL performs faster than RS is in the different delivery conditions used by the two methods. Indeed, considering how RS preferences are defined, it can be shown that $(\underline{\mathbf{p}} \not\succ_{RS} \mathbf{p_i}) \Rightarrow (\underline{\mathbf{p}} \not\succ_{SL} \mathbf{p_i})$, thus the delivery condition of RS is always more restrictive than that of SL.

Finally we present graphs where efficiency and quality of results can be observed together. Figure 11 shows how much we have to pay (in terms of sorted and random accesses, respectively) for each relevant object we retrieve. The graphs confirm previous results, in particular the superior performance of qualitative preferences, and also show that, starting with $k_{rel} \geq 50$, the reduced efficiency of RS with respect to SL is compensated by its superior effectiveness, which leads to a "per relevant object" cost of RS almost equal to that of SL.

Experiment 4: The aim of this final experiment is to compare iMPO (equipped with either SL or RS preferences) with the *median rank* criterion proposed in [15]. For an object o_i and m sub-queries, the



(a) (b)

Figure 11. Sorted accesses (a) and random accesses (b) vs no. of relevant retrieved objects (k_{rel})

median rank of o_i is defined as:

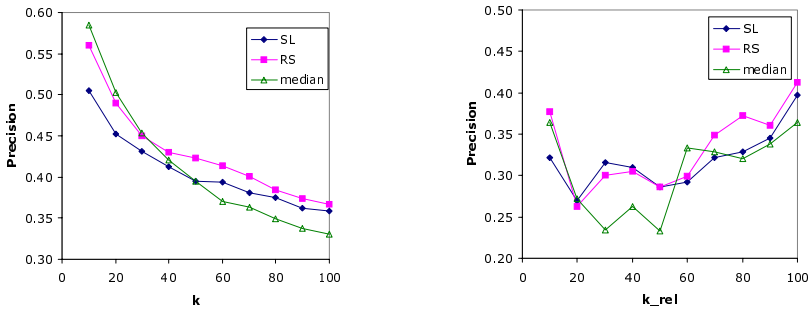
$$medrank(o_i) = median(rank(o_i, L_1), \ldots, rank(o_i, L_m)) \qquad (6)$$

Although in [15] this criterion was proposed as part of an (approximate) alternative to Euclidean high-dimensional nearest neighbor search, which also requires feature vectors to be projected onto m randomly-chosen lines, we consider it here since it represents an original integration rule.

In order to determine the best k objects according to *medrank*, we slightly modify iMPO, along the lines of the MEDRANK algorithm in [15]. In particular, the object that has the minimum value of *medrank* is the first one that is retrieved via sorted access on at least $\lfloor m/2 + 1 \rfloor$ ranked lists, the second best object is the second one seen on more than half of the lists, and so on [15]. Note that this way random accesses are not required at all.

Figure 12 (a) indeed shows that *medrank* results are competitive with those of SL and RS, at least when the number of retrieved objects is not too large. In spite of this, it has however to be observed that, since *medrank* definitely does not fit Definition 2, with this method *the relative order of any two objects does not depend only on the objects themselves, but also on other objects in the database*. More precisely, it is not possible to tell if, say, $medrank(o_i) < medrank(o_j)$ without also knowing which other objects are present. This is evident from Equation 6, since the rank of object o_i in list L_q, $rank(o_i, L_q)$, does not depend only on "how well" o_i matches sub-query Q_q but also on how other objects perform on the same sub-query. Intuitively, this might affect the "stability" of *medrank* results in front of database changes.

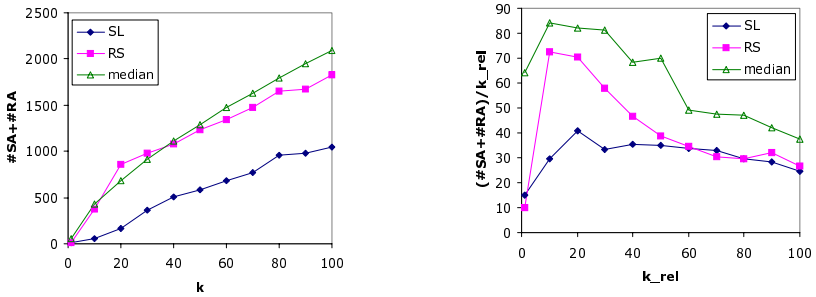
As to costs, Figure 13 shows the total number of accesses (i.e., sorted plus random) of the three alternatives. Again, the "per relevant object"



(a) (b)

Figure 12. Precision of SL, RS and median vs no. of retrieved objects (a) and no. of relevant retrieved objects (b)

costs of RS and SL (Figure 13 (b)) confirm the overall best behavior of qualitative preferences.



(a) (b)

Figure 13. Access costs of SL, RS, and median vs no. of retrieved objects (a) and no. of relevant retrieved objects (b)

Visual Example: We conclude this section by showing some actual results of the analyzed methods. The aim is to provide visual evidence of how changing the integration rule can indeed strongly influence what the user actually sees, even if evaluation of the underlying sub-queries always remains the same. In particular, in the example of Figure 14, which just shows the top 8 objects for a query in the semantic class "birds", SL and RS preferences return the same result set, with a 50% precision (4 relevant images out of 8), whereas avg and median only return 2 and 1 relevant images, respectively. Concerning access costs, for this specific query SL just requires to retrieve the top 4 objects from

each of the 4 lists (the query is split into 4 regions) to deliver the images shown. This grows to 21×4 sorted accesses for the RS preferences, which is however still much less than both the 436 accesses executed when avg is used and the 441 of *medrank*.

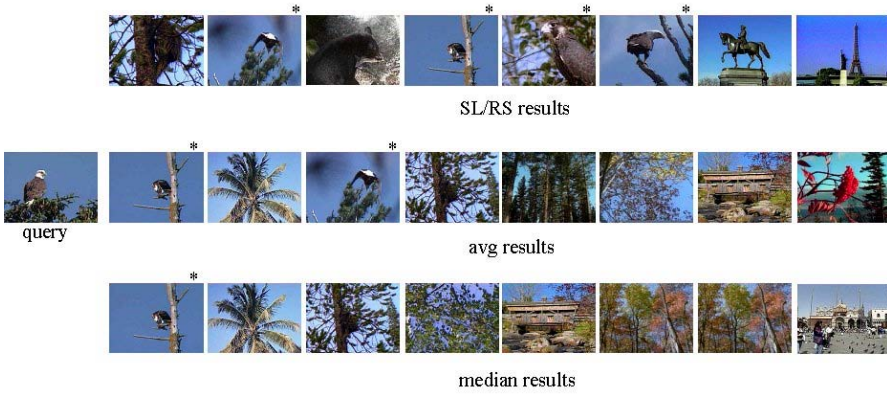


Figure 14. Visual results for a "bird" query image. Images in the same semantic class of the query are adorned with an asterisk (*)

As a final analysis, we measure how much the visual results in Figure 14, as well as those of other queries, are significantly different from those obtainable from a purely "random" ranking process. To this end we use the well-known Wilcoxon signed rank test [10] that consists in comparing the differences between a series of n paired observations, in order to determine whether the two series originate from a same underlying distribution. In our case we compare the precision values obtainable from RS preferences to those of a random sample of images, for $n = 20$ queries and $k = 50$ retrieved images. The result of the test is that the probability that the ranking induced by RS preferences is random is less than 10^{-5}, that is, impossible from a practical point of view. The same is true even for Skyline preferences, as well as for the avg and median rank integration criteria. This demonstrates the statistical significance of our results and dissipates any doubt that the precision values observed in our experiments are just an effect of causality.

6. Related Work

As far as we know, no other works has attempted to analyze the effects of using qualitative preferences for multimedia complex queries. Thus, in this section we limit to survey works that are related to ours for what concerns the query processing aspects, that is, integration algorithms.

Starting with the seminal paper by Fagin [14], several works have addressed the problem of how to efficiently compute the result of a complex query that is split into m sub-queries. Apart from those focused on answering top k queries using monotone scoring functions (see, e.g., [16, 9]), closer to our work are those algorithms that have been developed to deal with more complex types of user preferences. Along this direction [2] has proposed an algorithm for integrating results of sub-queries based on Skyline (i.e., Pareto) preferences. This has been generalized in [1] to scenarios in which several scoring functions are applied to sub-queries results, and then the values of such scoring functions are integrated using Pareto preferences. Although MPO and iMPO share with such algorithms a similar logic (all being based on a common access model, i.e., sorted and random accesses), there are indeed some important differences. First, the algorithms in [2] and [1] both return all and only the best matches of a query. As argued in Section 4.2 this is not always desirable, since too many or too few results may be returned to the user. A second major difference concerns the type of qualitative preferences supported by the algorithms. While iMPO (and MPO as well) can deal with any strictly monotone partially ordered preference relation, algorithms in [2] and [1] are of less general use. In particular, they are not designed to deal with prioritized preferences, such as RS.

7. Conclusions

In this paper we have analyzed the potentialities of using *qualitative* (rather than *quantitative*) preferences for the integration of the results of multiple multimedia sub-queries. In particular, we have focused on Skyline (Pareto) preferences and on their generalization to the case where one wants to set some priorities on the regions of the answer space (RS preferences). The latter have indeed shown to provide the best characterization of the distribution of the relevant objects of a query, as compared to Skyline preferences and to numerical scoring functions (namely, *min* and average).

For the purpose of efficient evaluation we have introduced two algorithms that generalize previous integration algorithms and analyzed their performance on a real-world image database. In particular, the iMPO algorithm can incrementally deliver its results and is suitable to answer top k queries. Using iMPO we have been able to show that with qualitative preferences one can considerably reduce the costs to be paid for obtaining a fixed number of relevant objects.

Our work opens new interesting lines of research. First, it would be interesting to apply qualitative preferences to other challenging tasks, such as classification of multimedia objects, for which scoring functions have been considered the only viable alternative. Second, qualitative preferences could also be profitably used for the evaluation of subqueries, thus generalizing the common approach requiring a distance metrics to compare objects' features.

References

[1] Wolf-Tilo Balke and Ulrich Güntzer. Multi-Objective Query Processing for Database Systems. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB'04)*, pages 936–947, Toronto, Canada, September 2004.

[2] Wolf-Tilo Balke, Ulrich Güntzer, and Jason Xin Zheng. Efficient Distributed Skylining for Web Information Systems. In *Proceedings of the 6th International Conference on Extending Database Technology (EDBT'04)*, pages 256–273, Heraklion, Crete, March 2004.

[3] Ilaria Bartolini, Paolo Ciaccia, Vincent Oria, and M. Tamer Özsu. Integrating the Result of Multimedia Queries Using Qualitative Preferences. Technical Report IEIIT-BO-06-04, IEIIT, April 2004.

[4] Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. A Sound Algorithm for Region-Based Image Retrieval Using an Index. In *Proceedings of the 4th International Workshop on Query Processing and Multimedia Issue in Distributed Systems (QPMIDS'00)*, pages 930–934, Greenwich, London, UK, September 2000.

[5] Ilaria Bartolini, Paolo Ciaccia, and Florian Waas. FeedbackBypass: A New Approach to Interactive Similarity Query Processing. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, pages 201–210, Rome, Italy, September 2001.

[6] Michèle Basseville. Distance Measures for Signal Processing and Pattern Recognition. *European Journal of Signal Processing*, 18(4):349–369, 1989.

[7] Klemens Böhm, Michael Mlivoncic, Hans-Jörg Schek, and Roger Weber. Fast Evaluation Techniques for Complex Similarity Queries. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB'01)*, pages 211–220, Rome, Italy, September 2001.

[8] S. Börzsönyi, D. Kossmann, and K. Stocker. The Skyline Operator. In *Proceedings of the 17th International Conference on Data Engineering (ICDE'01)*, pages 421–430, Heidelberg, Germany, April 2001.

[9] Nicolas Bruno, Luis Gravano, and Amélie Marian. Evaluating Top-k Queries over Web-Accessible Databases. In *Proceedings of the 18th International Conference on Data Engineering (ICDE'02)*, pages 369–382, San Jose, California, USA, February 2002.

[10] George C. Canavos. *Applied Probability and Statistical Methods*. Little, Brown & Co., Toronto, Canada, 1984.

[11] Jan Chomicki. Querying with Intrinsic Preferences. In *Proceedings of the 8th International Conference on Extending Database Technology (EDBT'02)*, pages 34–51, Prague, Czech Republic, March 2002.

[12] Paolo Ciaccia, Marco Patella, and Pavel Zezula. Processing Complex Similarity Queries with Distance-based Access Methods. In *Proceedings of the 6th International Conference on Extending Database Technology (EDBT'98)*, pages 9–23, Valencia, Spain, March 1998.

[13] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[14] Ronald Fagin. Combining Fuzzy Information from Multiple Systems. In *Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'96)*, pages 216–226, Montreal, Canada, June 1996.

[15] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Efficient Similarity Search and Classification via Rank Aggregation. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD'03)*, pages 301–312, San Diego, California, USA, June 2003.

[16] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal Aggregation Algorithms for Middleware. In *Proceedings of the 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'01)*, pages 216–226, Santa Barbara, California, USA, May 2001.

[17] Peter C. Fishburn. Preference Structures and Their Numerical Representations. *Theoretical Computer Science*, 217(2):359–383, 1999.

[18] Werner Kießling. Foundations of Preferences in Database Systems. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02)*, pages 311–322, Hong Kong, China, August 2002.

[19] Michael Ortega, Yong Rui, Kaushik Chakrabarti, Kriengkrai Porkaew, Sharad Mehrotra, and Thomas S. Huang. Supporting Ranked Boolean Similarity Queries in MARS. *IEEE Transactions on Knowledge and Data Engineering*, 10(6):905–925, 1998.

[20] Kriengkrai Porkaew, Sharad Mehrotra, and Michael Ortega. Query Reformulation for Content Based Multimedia Retrieval in MARS. In *Proceedings of the International Conference on Multimedia Computing and Systems (ICMCS'99)*, volume 2, pages 747–751, Florence, Italy, June 1999.

[21] Yong Rui, Thomas S. Huang, Michael Ortega, and Sharad Mehrotra. Relevance Feedback: A Power Tool for Interactive Content-Based Image Retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):644–655, 1998.

[22] Riccardo Torlone and Paolo Ciaccia. Which Are My Preferred Items? In *AH2002 Workshop on Recommendation and Personalization in eCommerce (RPeC02)*, pages 1–9, Malaga, Spain, May 2002.