Taking Care of Vagueness and User Preferences for Effective Similarity Queries on Multimedia Data

Danilo Montesi[1] and Wilma Penzo[2]

[1] DSI, University of Milano, Italy
montesi@dsi.unimi.it
[2] DEIS CSITE-CNR, University of Bologna, Italy
wpenzo@deis.unibo.it

Abstract. Due to the complex nature of multimedia data and the inherent difficulty of classifying objects in a clear and precise form, traditional exact matching query processing techniques turn out to be too restrictive and do not lead to satisfactory results in terms of effectiveness. In this paper we present an algebraic framework, SAME^W, in order to model and query multimedia data in a flexible way. As a starting point, we assume the more general case where "imprecision" is already present at the data level, typically because of the ambiguous nature of multimedia objects' content. Queries on such complex data are carried out by means of similarity and fuzzy predicates which prove to be effective ingredients to return interesting results matching the user requirements. A further basic feature of SAME^W is that it allows user preferences, expressed in the form of *weights*, to be specified so as to assign different importance to predicates in queries. Moreover, in SAME^W the semantics of operators is deliberately left unspecified in order to better adapt to specific scenarios.

1 Introduction

Due to the complex nature of multimedia data and the inherent difficulty of classifying objects in a clear and precise form, traditional exact matching query processing techniques turn out to be too restrictive, and many systems now exist that allow users to issue queries where some form of "imprecision" is allowed. For instance, in the QBIC system [FSN+95] images can be searched on the basis of their "similarity" to a target one, according to color, texture, and shape of embedded objects. Since MM queries can lead to (very) high response times, many efforts have been spent in order to devise access methods able to efficiently deal with complex features [CPZ97]. This line of research has been somewhat complemented by activity aiming to provide users with a full-fledged query language able to express complex similarity queries [NRT99]. Although the processing of such complex queries has been the subject of some recent works [ABSS98,CPZ98,Fag96,MT99], a full understanding of the implications (both at a formal and at a system level) of similarity query processing is still lacking. In particular, several important issues have only partially been addressed, such as models able to capture the "essential" aspects of MM objects needed by similarity queries, the impact of "user preferences" on query processing, query equivalence, and so on. Furthermore, contributions to above issues typically consider ad-hoc scenarios and/or completely ignore the other coordinates of the problem, thus resulting in a set of difficult-to-integrate recipes.

In this work we address several of the above points in a unified algebraic framework. We have deliberately chosen to "start simple" from the modeling point of view, in order to better focus on those aspects which are peculiar to similarity query processing. Thus, we consider a (extended) relational framework which takes into account the two major sources of "imprecision" arising when querying MM databases [SS98]: 1) imprecision of *classification* of MM data, and 2) imprecision in the *matching of features* that characterize the content of MM objects. As to the first point we rely

on basic concepts from *fuzzy set theory*, and allow representation of "vague classification" both at tuple and at attribute level (Section 2). This reflects the fact that in some cases imprecision characterizes an object as a whole, whereas in others it only affects some of its attributes. We then consider the presence of "weights" in the queries, so as to assign different relevance to the required conditions, in order to better adapt to user preferences. In Section 3 we introduce a "similarity algebra", called SAME^W,[1] which extends relational algebra in a conservative way and incorporates the use of weights in most of its operators. We show how complex similarity queries can be easily expressed in SAME^W(Section 3) and how equivalence rules can be exploited for the purpose of query rewriting and optimization (Section 4). Finally, we briefly discuss related work and conclude.

2 A Flexible Approach

The SAME^{W}data model extends the relational one in that it allows both *fuzzy attributes* and *fuzzy relations*. This results in a more flexible approach to deal with imprecision that may occur, for instance, when multimedia data is modeled, because of the well-known intrinsic complexity of non-textual information, as well as when data has to be arranged according to classification purposes or specific (application) criteria.

As shown in [CMPT00], imprecision at the attribute level is captured by the notion of "fuzzy domain", whereas imprecision at the whole tuple level motivates the introduction of fuzzy relations[2]. Briefly, given a tuple t, a fuzzy attribute A in t is formed by two components A^v (the "value", that is a fuzzy set), and A^μ (the "score") which, intuitively, have the following meaning: "t fits A^v with score A^μ". Given a set of attributes X and a relation name R, a fuzzy relation r over $R(X)$ is characterized by a membership function μ_R which represents how much a given object (tuple) "fits" the concept expressed by $R(X)$[3]. Besides fuzzy attributes, our model also includes the concept of *feature attribute*, defined over complex domains like, say, `color_histogram`, which are required to represent feature values extracted from MM objects. For such attributes *similarity predicates* (rather than exact-matching) are the usual way to compare feature values (see Section 2.1). A further peculiarity of SAME^{W}is that it allows the use of weights to deal with user preferences when retrieving information of interest. In order to present such a functionality we rely on the results gained by Fagin and Wimmers [FW97] (see Section 2.2).

2.1 Dealing with Imprecision

SAME^{W}expressions can use ordinary (Boolean), *similarity*, and *fuzzy* predicates which can be combined into complex formulas with logical connectives respecting the syntax $f ::= p|f \wedge f|f \vee f|\neg f|(f)$, where f is a formula and p is a predicate. The *evaluation* of f on a tuple t is a score $s(f,t) \in \mathcal{S} = [0,1]$ which says how much t satisfies f. How $s(f,t)$ depends on (the evaluation on t of) the predicates in f is intentionally left unspecified, in order to achieve parametricity with respect to the semantics of logical operators, which can therefore be varied in order to better adapt to user and application requirements. $s(f,t)$ is computed by means of a so-called "scoring function" [Fag96], s_f, whose arguments are the scores, $s(p_i,t)$, of t with respect to the predicates in f, that is: $s(f(p_1,\ldots,p_n),t) = s_f(s(p_1,t),\ldots,s(p_n,t))$. A similarity predicate has either the form

[1] SAME^{W}stands for "Similarity Algebra for Multimedia Extended with Weights".

[2] Remind that a fuzzy set F over a "universe" U is a set characterized by a *membership function* $\mu_F : U \rightarrow \mathcal{S}$, where $\mu_F(x)$ is the *degree of membership* of x in F, also called "score" or "grade". In the following we will always consider a normalized *score domain* $\mathcal{S} = [0,1]$.

[3] The notation $t.\mu_R$ will be used with the same meaning of $\mu_R(t)$.

$A \sim v$, where A is a feature attribute, $v \in dom(A)$ is a constant, and $\sim$ is a *similarity operator*, or $A_1 \sim A_2$, where both A_1 and A_2 are over the same domain.[4] The evaluation of $p : A \sim v$ on t returns a score, $s(p,t) \in \mathcal{S}$, which says how much $t.A$ is similar to the value v. Then, the evaluation on t of a fuzzy predicate $q : A = w$, where w is a fuzzy set, is the score $s(q,t) = t.A^{\mu}$, if $t.A^{v} = w$, otherwise $s(q,t) = 0$. For fuzzy predicates of the form $q : A_1 = A_2$, $s(q,t) = 0$ if $t.A_1^{v} \neq t.A_2^{v}$, otherwise the score is computed as a "parametric conjunction" of the two membership degrees, that is, $s(q,t) = s_{\wedge}(t.A_1^{\mu}, t.A_2^{\mu})$, where $s_{\wedge}$ denotes the AND scoring function. For the sake of definiteness, in the following we restrict our focus on the class $\mathcal{F}$ of scoring functions corresponding to fuzzy *t-norms* and *t-conorms* [KY95,Fag96], for which the AND ($\wedge$) and OR ($\vee$) operators are both associative and commutative, and, together with the NOT ($\neg$) operator, satisfy *boundary* and *monotonicity* conditions [CMPT00]. For instance, $\mathcal{FS}$ (fuzzy standard) and $\mathcal{FA}$ (fuzzy algebraic) [KY95] semantics are given by the following set of rules:

	$\mathcal{FS}$	$\mathcal{FA}$
$s(f_1 \wedge f_2, t)$	$\min(s(f_1,t), s(f_2,t))$	$s(f_1,t) \cdot s(f_2,t)$
$s(f_1 \vee f_2, t)$	$\max(s(f_1,t), s(f_2,t))$	$s(f_1,t) + s(f_2,t) - s(f_1,t) \cdot s(f_2,t)$
$s(\neg f, t)$	$1 - s(f,t)$	$1 - s(f,t)$

2.2 Dealing with User Preferences: Weights

With a non-Boolean semantics, it is quite natural and useful to give the user the possibility to assign a different relevance to the conditions he states to retrieve tuples. Such "user preferences" can be expressed by means of *weights* as shown in [FW97]. To this end, we apply the weighted version $s_{f\Theta}$ of any scoring function s_f for a formula f on a set of predicates $p_1, \ldots, p_n$. More in details, a vector of weights $\Theta = [\theta_1, \ldots, \theta_n]$ is considered in such a way that $s_{f\Theta}$ reduces to s_f when all the weights are equal, $s_{f\Theta}$ does not depend on $s(p_i,t)$ when $\theta_i = 0$, and $s_{f\Theta}$ is a continuous function of the weights, for each fixed set of argument scores. Let $x_i = s(p_i,t)$ denote the score of t with respect to p_i, and assume without loss of generality $\theta_1 \geq \theta_2 \geq \ldots \geq \theta_n$, with $\theta_i \in [0,1]$ and $\sum_i \theta_i = 1$. Then, Fagin and Wimmers' formula is:

$$s_{f\Theta_f}(x_1, \ldots, x_n) = (\theta_1 - \theta_2) \cdot x_1 + 2 \cdot (\theta_2 - \theta_3) \cdot s_f(x_1, x_2) + \cdots + n \cdot \theta_n \cdot s_f(x_1, \ldots, x_n) \qquad (1)$$

Although above formula is usually used to weigh the predicates appearing in a (selection) formula, our position is that *whenever scores have to be "combined", then a weighting should be allowed.* For instance, if we take the union of two relations, it might be reasonable to require that tuples in the first relation are "more important" than tuples in the second one. A meaningful example is when we want to integrate results from different search engines, but we trust more one than the other. Accordingly, most of the SAME^W operators[5] that compute new tuples' scores can use weights.

3 The SAME^W Algebra

Basic operators of SAME^W conservatively extend those of RA in such a way that, if no "imprecision" is involved in the evaluation of an expression, the semantics of RA applies (see Theorem 1). Genericity with respect to different semantics is achieved by defining SAME^W operators in terms

[4] This is a simplification. It suffices that the domains are compatible for '$\sim$' to be well-defined.

[5] We only leave out Difference, because we were not able to conceive any meaningful "weighted Difference" query. As to Projection, since the duplicate tuples to be combined together are not a priori known, it is not possible to assign weights to them.

of the (generic) scoring functions of the logical operators. Thus, if a given semantics is adopted for formulas, the same is used by SAME^W operators, which avoids counter-intuitive phenomena and preserves many RA equivalence rules. As an example, the semantics of Union ($\cup$) is based on that of the OR ($\vee$) operator.

In the following, $E(X)$ denotes an expression with schema X, and $e = E[db]$ is the fuzzy set of tuples with schema X obtained by evaluating $E(X)$ over the current database db. We say that a tuple t belongs to e ($t \in e$) iff $t.\mu_E > 0$ holds. Two tuples t_1 and t_2 with attributes X are *equal* iff $t_1[A_i] = t_2[A_i]$ holds for each $A_i \in X$. In case of fuzzy attributes, tuple equality thus requires that also the attributes' grades are the same. Two relations e_1 and e_2 are equal iff: 1) they consist of the same set of tuples, and 2) $\forall t_1 \in e_1, \forall t_2 \in e_2 : t_1 = t_2 \Rightarrow t_1.\mu = t_2.\mu$. Thus, besides tuple equality also the tuple scores have to be the same.

We start by extending "traditional" operators of RA, and then introduce new operators which have no direct counterpart in RA. In order to show the potentialities and flexibility of SAME^W we carry on a simple example which refers to a *biometric DB* using faces and fingerprints to recognize the identity of a person. Stored data include extracted features relevant for identification,[6] and modeled by the `FaceFV` and `FP_FV` attributes, respectively. Because of the huge size of biometric databases, a viable way to improve performance is, at face and fingerprint acquisition time, to *classify* them with respect to some predefined classes. As to fingerprints, as demonstrated in [LMM97], a "continuous" classification approach, where a fingerprint is assigned with some degree to many (even all) classes, can perform better than an approach based on "exclusive" classification. As to faces, we consider that the `Chin` and the `Hair` are also preventively classified.

Our simplified biometric DB consists of the following relations, where the '$*$' denotes fuzzy attributes and relations that can have fuzzy instances, and primary keys are underlined. The `Freq` attribute is the relative frequency of a fingerprint class.[7] This can be computed by considering the *scalar cardinality* (also called the *sigma count* [KY95]) of the fuzzy set corresponding to the fingerprint class. A partial instance is shown in Fig. 1.

`Persons(`<u>`PId`</u>`,Name)`
`Faces(`<u>`PId`</u>`,FaceFV,Chin`*`,Hair`*`)`
`FingerPrints(`<u>`FPId`</u>`,PId,FingerNo,FP_FV)`
`FPClasses(`<u>`Class`</u>`,Freq)`
`FPType`*`(`<u>`FPId,Class`</u>`)`

Selection (σ). The Selection operator applies a formula f to the tuples in e and filters out those which do not satisfy f. The novel point here is that, as an effect of f and of weights, *the grade of a tuple t can change*. Weights can be used for two complementary needs: In the first case, they weigh the importance of predicates in f, as in [FW97], thus leading to use the scoring function $s_{f_{\Theta_f}}$ in place of s_f.[8] In the second case they are used to perform a *weighted conjunction*, $s^{\Theta}_{\wedge}$, between the score computed by f and the "input" tuple score, $t.\mu_E$. This determines the new tuple score, $t.\mu$:

$$\sigma^{\Theta}_{f_{\Theta_f}}(e) = \{t \mid t \in e \wedge t.\mu = s^{\Theta}_{\wedge}(s(f_{\Theta_f}, t), t.\mu_E) > 0\} \tag{2}$$

Example 1. Retrieve those persons who have black hair, and whose facial features match the ones (`inFace`) given in input, by trusting more hair classification (weight 0.7) than feature matching

[6] For fingerprints these can be "directional vectors", positions of "minutiae", etc. For faces, position of eyes, nose, etc., can be considered.

[7] Class names are among those adopted by NIST (U.S. National Institute of Standards and Technology).

[8] When using weights, f is restricted to be either a conjunction or a disjunction of predicates by Fagin's definition [Fag96].

Persons

PId	Name
P00001	John
P00002	Mary
P00003	Bill
P00004	Jack
P00005	Susan

FPClasses

Class	Freq
Arch	3.7%
LeftLoop	33.8%
RightLoop	31.7%

Faces

PId	FaceFV	Chin	Hair
P00001	FFV0001	pointed:0.74	black:0.87
P00002	FFV0002	rounded:0.65	brown:0.75
P00003	FFV0003	pointed:0.93	brown:0.84
P00004	FFV0004	pointed:0.58	brown:0.73
P00005	FFV0005	rounded:0.83	brown:0.92

FingerPrints

FPId	PId	FingerNo	FP_FV
FP0001	P00001	1	FPFV0001
FP0002	P00001	2	FPFV0002
FP0003	P00001	3	FPFV0003
FP0011	P00002	1	FPFV0011
FP0015	P00002	5	FPFV0015
FP0017	P00002	7	FPFV0017
FP0025	P00004	3	FPFV0025

FPType

FPId	Class	μ
FP0001	Arch	0.65
FP0001	RightLoop	0.25
FP0003	Arch	0.58
FP0003	LeftLoop	0.95
FP0025	Arch	0.45
FP0025	LeftLoop	0.20

Fig. 1. An instance of the biometric database

(weight 0.3):

$$\sigma_{(Hair='black')^{0.7} \wedge (FaceFV \sim \mathtt{inFace})^{0.3}}(Faces)$$

Consider the similarity table $\sim$`inFace` shown in Figure 2. The final score of a tuple t is obtained

$\sim$ inFace

FaceFV	score
FFV0001	0.60
FFV0002	0.84
FFV0003	0.33
FFV0004	0.72
FFV0005	0.58

$\sim$ inFP

FP_FV	score
FPFV0001	0.72
FPFV0002	0.48
FPFV0003	0.43
FPFV0011	0.84
FPFV0015	0.38
FPFV0017	0.55
FPFV0025	0.42

Fig. 2. Similarity tables for fingerprints and facial features

by combining the scores of both Selection predicates and the initial score of the tuple (this is 1, since `Faces` is a crisp relation). For instance, if the $s_\wedge = \min$ scoring function is used, the score of the unique resulting tuple is computed as:[9]

$$t.\mu = (0.7 - 0.3) \cdot s(p_1, t) + 2 \cdot 0.3 \cdot s_\wedge(s(p_1, t), s(p_2, t)) = 0.4 \cdot 0.87 + 0.6 \cdot \min(0.87, 0.60) = 0.708$$

Projection (π). As in RA, the Projection operator removes a set of attributes and then eliminates duplicate tuples. Projection can also be used to *discard* scores, both of fuzzy attributes and of the whole tuple. In this case, however, in order to guarantee consistency of subsequent operations, such scores are simply set to 1, so that they can still be referenced in the resulting schema. This captures the intuition that if we discard, say, the tuples' scores, then the result is a crisp relation, that is, a fuzzy relation whose tuples all have score 1.

Formally, let e be a relation with schema $E(X)$, $Y \subseteq X$, and V a set of *v-annotated* fuzzy attributes, $V = \{A_i^v\}$, where V contains exactly those fuzzy attributes for which scores are to be

[9] We omit the score of the original crisp tuple since it equals to 1 and it does not influence the computation.

discarded. Note that V can include A_i^v only if $A_i \in X - Y$. Finally, let F stand for either μ or the empty set. Then, the projection of e over YVF is a relation with schema YW, where if $A_i^v \in V$ then $A_i \in W$, defined as follows:

$$\pi_{YVF}(e) = \{t[YW] | \exists t' \in e : t[YV] = t'[YV] \wedge \forall A_i^v \in V : t.A_i^\mu = 1 \\ \wedge\, t.\mu = s_\vee\{t''.\mu_E | t''[YV] = t[YV]\} \text{ if } F = \mu, \text{ otherwise } t.\mu = 1\} \quad (3)$$

Thus, tuples' scores are discarded (i.e. set to 1) when $F = \emptyset$, whereas they are preserved when $F = \mu$. In the latter case, new scores are computed by considering the "parametric disjunction", $s_\vee$, of the scores of all duplicate tuples with the same values for YV (see Example 5).

Union ($\cup$). In SAME^{W}the Union is an n-ary operator,[10] which, given n relations e_i with schemas $E_i(X)$, computes the score of a tuple t as a *weighted disjunction*, $s_\vee^\Theta(t.\mu_{E_1}, \ldots, t.\mu_{E_n})$, with $\Theta = [\theta_1, \ldots, \theta_n]$, of the input tuples' scores, that is:

$$\cup^\Theta(e_1, \ldots, e_n) = \{t \mid (t \in e_1 \vee \ldots \vee t \in e_n) \wedge t.\mu = s_\vee^\Theta(t.\mu_{E_1}, \ldots, t.\mu_{E_n}) > 0\} \quad (4)$$

Note that, because of the presence of weights, Union is not associative anymore. This implies that the n-ary Union cannot be defined in terms of $n-1$ binary unions, as it happens in RA.

Example 2. Retrieve Ids and scores of fingerprints belonging to class 'Arch' (weight 0.6) and Ids and scores of fingerprints belonging to class 'LeftLoop' (weight 0.4).
This can be expressed as follows[11]:

$$\pi_{FPId,\mu}(\sigma_{Class='Arch'}(FPType)) \cup^{[0.6,0.4]} \pi_{FPId,\mu}(\sigma_{Class='LeftLoop'}(FPType))$$

and the resulting tuples are:

FPId	score
FP0001	0.65
FP0003	0.876
FP0025	0.45

where, for instance, the score of the second tuple has been computed as (where the $s_\vee$ = max scoring function has been used):

$$t.\mu = (0.6 - 0.4) \cdot t.\mu_{E_1} + 2 \cdot 0.4 \cdot s_\vee(t.\mu_{E_1}, t.\mu_{E_2}) = 0.2 \cdot 0.58 + 0.8 \cdot \max(0.58, 0.95) = 0.876$$

Join ($\bowtie$). Also the weighted (natural) Join is an n-ary operator, where, given n relations e_i with schemas $E_i(X_i)$, $\forall i = 1, ..., n$ and where the score of a result tuple t is a *weighted conjunction*, $s_\wedge^\Theta(t_1.\mu_{E_1}, \ldots, t_n.\mu_{E_n})$, of the scores of matching tuples:

$$\bowtie^\Theta (e_1, \ldots, e_n) = \{t[X_1 \ldots X_n] \mid \exists t_1 \in e_1, \ldots, \exists t_n \in e_n : \\ t[X_1] = t_1[X_1] \wedge \ldots \wedge t[X_n] = t_n[X_n] \wedge t.\mu = s_\wedge^\Theta(t_1.\mu_{E_1}, \ldots, t_n.\mu_{E_n}) > 0\} \quad (5)$$

Example 3. Retrieve those persons with an 'Arch' fingerprint and with a 'pointed' chin, giving these conditions weights 0.6 and 0.4, respectively.
This can be expressed by means of a weighted Join, where a 0 weight is used for the `FingerPrints` relation on which no predicates are specified:

$$\bowtie^{[0.6,0.4,0]} (\sigma_{Class='Arch'}(FPType), \sigma_{Chin='pointed'}(Faces), FingerPrints)$$

where the resulting tuples, are:

[10] We also use the infix notation, $E_1 \cup^{[\theta_1,\theta_2]} E_2$, when only two operands are present.

[11] Note that this is not equivalent to $\pi_{FPId,\mu}(\sigma_{(Class='Arch')^{0.6} \vee (Class='LeftLoop')^{0.4}}(FPType))$ since weights are assigned to predicates on `Class` which is a crisp attribute, whereas the operands of the above Union are fuzzy relations.

FPId	Class	PId	FaceFV	Chin	Hair	FingerNo	FP_FV	μ
FP0001	Arch	P00001	FFV0001	pointed:0.74	black:0.87	1	FPFV0001	0.65
FP0003	Arch	P00001	FFV0001	pointed:0.74	black:0.87	3	FPFV0003	0.58
FP0025	Arch	P00004	FFV0004	pointed:0.58	brown:0.73	3	FPFV0025	0.45

where, for instance, the score of the third tuple has been computed as (with $s_\wedge = \min$):

$$t.\mu = (0.6 - 0.4) \cdot t.\mu_{E_1} + 2 \cdot (0.4 - 0) \cdot s_\wedge(t.\mu_{E_1}, t.\mu_{E_2}) + 3 \cdot 0 \cdot s_\wedge(t.\mu_{E_1}, t.\mu_{E_2}, t.\mu_{E_3}) =$$
$$= 0.2 \cdot 0.45 + 0.8 \cdot \min(0.45, 0.58) + 3 \cdot 0 \cdot \min(0.45, 0.58, 1) = 0.45$$

Difference $(-)$. Given relations e_1 and e_2 with schemas $E_1(X)$ and $E_2(X)$, respectively, their Difference is defined as:

$$e_1 - e_2 = \{t \,|\, t \in e_1 \wedge t.\mu = s_\wedge(t.\mu_{E_1}, s_\neg(t.\mu_{E_2})) > 0\} \tag{6}$$

Example 4. Retrieve the Ids and the scores of fingerprints of persons having a 'pointed' chin, but not belonging to the 'Arch' class.
The corresponding SAME^W expression is:

$$\pi_{FPId,\mu}(\sigma_{Chin='pointed'}(Faces) \bowtie FingerPrints) - \pi_{FPId,\mu}(\sigma_{Class='Arch'}(FPType))$$

and the resulting tuples are:

FPId	score
FP0001	0.35
FP0002	0.74
FP0003	0.42
FP0025	0.55

where, for instance, the score of the first tuple has been computed as (where the $s_\wedge = \min$ scoring function has been used):

$$t.\mu = s_\wedge(t.\mu_{E_1}, s_\neg(t.\mu_{E_2})) = \min(0.74, 1 - 0.65) = 0.35$$

Renaming (ρ). The Renaming operator is as in RA, thus we do not repeat its definition here.
The following result proves that the "RA-fragment" of SAME^W, that is, SAME^W restricted to the above operators, is indeed a conservative extension of relational algebra [CMPT00].

Theorem 1. *Let E be a SAME^W expression that does not use weights and that includes only operators in $\{\sigma, \pi, \cup, \bowtie, -, \rho\}$. If E does not use similarity operators, the database instance db is crisp and the semantics of the scoring functions is in $\mathcal{F}$, then $E[db] = E_{RA}[db]$, the latter being evaluated with the Boolean semantics of RA.*

The two new operators introduced in SAME^W are the *Top* and the *Cut*.

Top (τ). The Top operator retrieves the first k (k is an input parameter) tuples of a relation e, according to a *ranking criterion*, as expressed by a ranking function g. If weights are used to rank tuples according to g_{Θ_g}, then g has to be a formula of predicates over the schema of e.[12] If e has no more than k tuples, then $\tau^k_{g_{\Theta_g}}(e) = e$, otherwise:

$$\tau^k_{g_{\Theta_g}}(e) = \{t \,|\, t \in e \wedge s(g_{\Theta_g}, t) > 0 \wedge |\tau^k_{g_{\Theta_g}}(e)| = k \wedge \forall t \in \tau^k_{g_{\Theta_g}}(e) :$$
$$\nexists\, t' : t' \in e \wedge t' \notin \tau^k_{g_{\Theta_g}}(e) \wedge g_{\Theta_g}(t') > g_{\Theta_g}(t)\} \tag{7}$$

with ties arbitrarily broken. When g is omitted, the *default* ranking criterion, based on the score of tuples, applies, thus the k tuples with the highest scores are returned.

[12] If "bottom" tuples are needed, the *ranking directive* $<$ can be used, written $g_{\Theta_g,<}$.

*Example 5. Retrieve the chin and the final score of the three best matching tuples of persons who have brown hair (weight 0.7) and facial features (weight 0.3) matching those given in input (**inFace**):*

$$\pi_{Chin^v,\mu}(\tau^3(\sigma_{(Hair='brown')^{0.7}\wedge(FaceFV\sim\texttt{inFace})^{0.3}}(Faces)))$$

The tuples corresponding to the inner selection, and the final result obtained after Top and Projection, are shown in the following two tables, respectively:

PId	FaceFV	Chin	Hair	μ
P00002	FFV0002	rounded:0.65	brown:0.75	0.75
P00003	FFV0003	pointed:0.93	brown:0.84	0.534
P00004	FFV0004	pointed:0.58	brown:0.73	0.724
P00005	FFV0005	rounded:0.83	brown:0.92	0.716

Chin	μ
pointed	0.724
rounded	0.75

where the score of the second tuple of the final result (0.75) has been computed by assuming that the $s_\vee$ scoring function used by the Projection operator is the max function ($\max\{0.75, 0.716\}$, because tuples corresponding to PIds P00002 and P00005 belong to the intermediate result of the Top operator and have the same Chin value, i.e. rounded).

Cut (γ). The Cut operator "cuts off" those tuples which do not satisfy a formula g, that is:

$$\gamma_g(e) = \{t \mid t \in e \wedge s(g,t) > 0 \wedge t.\mu = t.\mu_E > 0\} \tag{8}$$

Unlike Selection, Cut *does not change tuples' scores*. Thus, if g includes non-Boolean predicates, the two operators would behave differently. However, the major reason to introduce Cut is the need of expressing *(threshold) conditions on tuples' scores*, e.g. $\mu > 0.6$. Such a predicate cannot be part of a Selection, since it does not commute with others. This is also to say that the expressions $\gamma_{\mu>0.6}(\sigma_f(E))$ and $\sigma_f(\gamma_{\mu>0.6}(E))$ are *not* equivalent. Indeed, the first expression is *contained* in the second one, that is:

$$\gamma_{\mu>\alpha}(\sigma_f(E)) \sqsubseteq \sigma_f(\gamma_{\mu>\alpha}(E)) \tag{9}$$

*Example 6. Given in input the two values **inFP** and **inFace**, retrieve the tuples of those persons having a corresponding combined match on fingerprints (weight 0.4) and facial features (weight 0.6) and an overall score greater or equal to 0.55:*

$$\gamma_{\mu\geq 0.55}(\sigma_{FaceFV\sim\texttt{inFace}}(Faces) \bowtie^{[0.6,0.4]} \sigma_{FP\_FV\sim\texttt{inFP}}(FingerPrints))$$

Consider the similarity table $\sim$inFP shown in Figure 2. The Join produces the following output:

PId	FaceFV	Chin	Hair	FPId	FingerNo	FP_FV	μ
P00001	FFV0001	pointed:0.74	black:0.87	FP0001	1	FPFV0001	0.60
P00001	FFV0001	pointed:0.74	black:0.87	FP0002	2	FPFV0002	0.504
P00001	FFV0001	pointed:0.74	black:0.87	FP0003	3	FPFV0003	0.464
P00002	FFV0002	rounded:0.65	brown:0.75	FP0011	1	FPFV0011	0.84
P00002	FFV0002	rounded:0.65	brown:0.75	FP0015	5	FPFV0015	0.472
P00002	FFV0002	rounded:0.65	brown:0.75	FP0017	7	FPFV0017	0.608
P00004	FFV0004	pointed:0.58	brown:0.73	FP0025	3	FPFV0025	0.48

then reducing to three final tuples which satisfy the cut-off condition:

PId	FaceFV	Chin	Hair	FPId	FingerNo	FP_FV	μ
P00001	FFV0001	pointed:0.74	black:0.87	FP0001	1	FPFV0001	0.60
P00002	FFV0002	rounded:0.65	brown:0.75	FP0011	1	FPFV0011	0.84
P00002	FFV0002	rounded:0.65	brown:0.75	FP0017	7	FPFV0017	0.608

4 Query Rewriting in SAME^W

The introduction of fuzzy attributes and fuzzy relations, as well as the presence of weights in user queries, inevitably introduces new aspects to be considered when reasoning about query rewriting and evaluation. This is mainly due to the different semantics all the operators have with respect to the standard relational algebra, as well as to the introduction of new operators.

Reminding that two relations e_1 and e_2 are equal iff: 1) they consist of the same set of tuples, and 2) $\forall t_1 \in e_1, \forall t_2 \in e_2 : t_1 = t_2 \Rightarrow t_1.\mu = t_2.\mu$, we can state that two expressions E_1 and E_2 are equivalent iff $E_1[db] = E_2[db]\ \forall db$. Basically, equivalence and containment rules in SAME^W can only partially rely on results from RA since, unless we consider the $\mathcal{FS}$ semantics, rules based on *idempotence* and/or *distributivity* of logical operators are no longer valid (e.g., $E \cup E \not\equiv E$ and $\sigma_f(E_1 \cup E_2) \not\equiv \sigma_f(E_1) \cup \sigma_f(E_2)$ in $\mathcal{FA}$). However, many opportunities for query rewriting are still left. As for weights, we will show that their presence leads to new interesting results, where numerical reasoning is strongly involved. For lack of space, here we discuss optimization issues related to a selected sample of such rules, focusing on Cut and Top operators and on the effect of weights. Complete proofs are given in [CMPT00]. In order to simplify the notation, when weights are not involved we understand their presence, thus writing, say, f in place of f_{Θ_f}.

4.1 The Role of Cut's

Consider the "canonical" Cut condition, $\gamma_{\mu>\alpha}$, abbreviated γ_α, applied to a Join. As for Selection in the relational algebra, we expect that the Cut operator can be "pushed down" the Join, since the Selection and the Cut operators can be considered equivalent. However, in SAME^W we have to carefully consider the case of a Cut applied to a *weighted* Join, i.e. when a different importance is assigned to tuples from different relations. This condition influences the criterion to cut off in advance those tuples which will not contribute to the result. In fact, as we will show in Example 7, also tuples with tuple's score lower than α can take part in the final result and, as a consequence, they cannot be discarded. Indeed, our first equivalence rule (Eq. 10) shows that a Cut γ_{α_i} can be applied to the i-th Join operand, where α_i depends on the values of weights (Eq. 11). The intuition is that, because of the weighting and the asymmetry of the weighted scoring function, also tuples with low tuple scores, if joined with tuples with a properly high score and having assigned a high weight, can contribute to a Join tuple which satisfies the external Cut condition, thus appearing in the result. Thus, in order to guarantee relevant data is not lost, thresholds of pushed down Cut's have to be relaxed for less important Join operands. Assuming $\theta_1 \geq \theta_2 \geq \ldots \geq \theta_n$, we have:

$$\gamma_\alpha(\bowtie^{[\theta_1,\ldots,\theta_n]}(E_1,\ldots,E_n)) \equiv \gamma_\alpha(\bowtie^{[\theta_1,\ldots,\theta_n]}(\gamma_{\alpha_1}(E_1),\ldots,\gamma_{\alpha_n}(E_n))) \tag{10}$$

where:

$$\alpha_i = \frac{\alpha - \sum_{j=1}^{i-1} j \cdot (\theta_j - \theta_{j+1})}{1 - \sum_{j=1}^{i-1} j \cdot (\theta_j - \theta_{j+1})} \qquad i \in [1..n] \tag{11}$$

Note that $\alpha_1 = \alpha$ and that $\alpha_i = \alpha$ when all the weights are equal, i.e. $\gamma_\alpha(\bowtie(E_1, E_2, \ldots, E_n)) \equiv \gamma_\alpha(\bowtie(\gamma_\alpha(E_1), \gamma_\alpha(E_2), \ldots, \gamma_\alpha(E_n)))$.

Example 7. Consider the following SAME^W expression:

$$\gamma_{0.6}(\bowtie^{[0.5,0.3,0.2]}(\sigma_{Chin='pointed'}(Faces), \sigma_{Class='Arch'}(FPType), FingerPrints))$$

The first and second Join operands, obtained by the Selection on `Faces` and `FPType`, respectively, are as follows:

PId	FaceFV	Chin	Hair	μ
P00001	FFV0001	pointed:0.74	black:0.87	0.74
P00003	FFV0003	pointed:0.93	brown:0.84	0.93
P00004	FFV0004	pointed:0.58	brown:0.73	0.58

FPId	Class	μ
FP0001	Arch	0.65
FP0003	Arch	0.58
FP0025	Arch	0.45

The third operand is just the `FingerPrints` relation. The result of the weighted Join is then (with $\mathcal{FS}$ semantics):

PId	FaceFV	Chin	Hair	FPId	Class	FingerNo	FP_FV	μ
P00001	FFV0001	pointed:0.74	black:0.87	FP0001	Arch	1	FPFV0001	0.668
P00001	FFV0001	pointed:0.74	black:0.87	FP0003	Arch	3	FPFV0003	0.612
P00004	FFV0004	pointed:0.71	brown:0.73	FP0025	Arch	3	FPFV0025	0.502

Finally, the external Cut discards the third tuple since it does not satisfy the threshold condition. Now, we show how the above SAME^W query can benefit of an optimized execution if expressed like in the right-hand side of Equation 10. In this case, it would have the form:

$$\gamma_{0.6}(\bowtie^{[0.5,0.3,0.2]} (\gamma_{0.6}(\sigma_{Chin=\text{`}pointed\text{'}}(Faces)), \gamma_{0.5}(\sigma_{Class=\text{`}Arch\text{'}}(FPType)), \gamma_{1/3}(FingerPrints)))$$

The first Cut reduces the cardinality of the intermediate result, i.e. the first operand of the weighted Join, since tuple corresponding to `PId P00004` has an overall score (0.58) lower than the required threshold. Similarly, the second Cut discards the last tuple of the second operand of the weighted Join (tuple score 0.45, threshold 0.5). On the other hand, the same tuples previously included in the Join would have been discarded later by the outer Cut. Thus the number of tuples involved in the Join are reduced by the inner Cuts. [13] Finally, it is worth to note that if the threshold of Cut operator for the second Join operand had been 0.6 instead of being relaxed to 0.5, the second tuple would have been discarded, thus losing the corresponding tuple of the final result.

A similar rule applies when the Cut has the form $\gamma_{\mu<\alpha}$ and follows a weigthed Union:

$$\gamma_{\mu<\alpha}(\cup^{[\theta_1,,\ldots,\theta_n]}(E_1,\ldots,E_n)) \equiv \gamma_{\mu<\alpha}(\cup^{[\theta_1,\ldots,\theta_n]}(\gamma_{\mu<\alpha_1}(E_1),\ldots,\gamma_{\mu<\alpha_n}(E_n))) \tag{12}$$

where $\alpha_i = \alpha/(1-\sum_{j=1}^{i-1} j\cdot(\theta_j-\theta_{j+1}))$, and $\alpha_i = \alpha$ if weights are not used.
Now, let consider the case of a sequence of canonical Cut's and Selection's applied to a given expression E:

$$\gamma_{\alpha_1}(\sigma_{f_1}(\gamma_{\alpha_2}(\sigma_{f_2}(\ldots\gamma_{\alpha_n}(\sigma_{f_n}(E)))))) \tag{13}$$

The intuition is that after each Selection σ_{f_i} tuples are required to have a score higher than α_i.
It is worth to note that, when $\alpha_1 \geq \alpha_i \quad \forall i \in [2..n]$, in order to belong to the final result, a tuple must satisfy the α_1 threshold condition also at the intermediate steps. This is equivalent to say that intermediate Cut's, supposed to have lower thresholds than α_1, do not change the final set of tuples, and they can be eliminated. Thus, such a complex expression can be reduced, without loss of information, as the following rule shows:

$$\gamma_{\alpha_1}(\sigma_{f_1}(\gamma_{\alpha_2}(\sigma_{f_2}(\ldots\gamma_{\alpha_n}(\sigma_{f_n}(E)))))) \equiv \gamma_{\alpha_1}(\sigma_{f_1\wedge f_2\wedge\ldots\wedge f_n}(E)) \qquad \text{if } \alpha_1 \geq \alpha_i \quad \forall i \in [2..n] \tag{14}$$

This is due to the following basic property of t-norms.

[13] The third Cut on `FingerPrints` is uninfluential since the relation is crisp and all tuples satisfy the required condition.

Property 1. Let $s_\wedge$ be a (n-ary) t-norm and $\mathcal{S} = [0,1]$ be the scoring domain. Then, $\forall x_1, .., x_n \in \mathcal{S}$:

$$s_\wedge(x_1, \ldots, x_n) \leq \min(x_1, \ldots, x_n) \leq x_i \qquad \forall i \in [1..n] \tag{15}$$

Thus, if the final score has to be higher than α_1, also scores of single components have to be, thus concluding that lower Cut's do not influence the final result.

Let us now consider the case where we apply to a *crisp* relation $E[db]$ a "cheap" predicate, p_1, and a "costly" one, p_2, after which we Cut the result. If predicates are weighted, and $\theta_1 \geq \theta_2$, we can apply a Cut just after evaluating p_1, which can lead to considerable cost saving. The intuition is that, in order to belong to the final result a tuple must have a score higher than α also when evaluating the most important predicate p_1. In fact, due to (1) for the binary case, it has to be:

$$(\theta_1 - \theta_2) \cdot s(p_1, t) + 2 \cdot \theta_2 \cdot s_\wedge(s(p_1, t), s(p_2, t)) > \alpha \tag{16}$$

Then, due to (15) and since $\theta_1 + \theta_2 = 1$, also it has to be $s(p_1, t) \geq \alpha$, thus concluding that γ_α can be safely applied after filtering tuples by p_1. This also shows how weights on predicates can be transformed into weights on tuples' scores:

$$\gamma_\alpha(\sigma_{p_1^{\theta_1} \wedge p_2^{\theta_2}}(E)) \equiv \gamma_\alpha(\sigma_{p_2}^{[\theta_2, \theta_1]}(\gamma_\alpha(\sigma_{p_1}(E)))) \tag{17}$$

Note that above equivalence *does not* hold if p_1 is commuted with p_2, when $\theta_1 > \theta_2$ holds. In this case, indeed, a tuple t can satisfy $\sigma_{p_1^{\theta_1} \wedge p_2^{\theta_2}}(E)$ but not $\sigma_{p_1}^{[\theta_1, \theta_2]}(\sigma_{p_2}(E))$. In particular, this is the case when $s(p_2, t) = 0$. In fact, in this case a tuple t which cannot result from the latter expression, can instead belong to the former one if a sufficiently high score is reached from the evaluation of p_1, and if weights are rather unbalanced, so that their difference is somewhat high.

4.2 Handling Top's

Turning to the Top operator, consider the case where the ranking criterion, g_{Θ_g} (or simply g) *does not* refer to tuples' scores. If no ties, according to g, occur in $E_i[db]$ ($i \in [1..n]$), then it is safe to apply a Top to each operand of a weighted Union, that is:

$$\tau_g^k(\cup^{[\theta_1, \ldots, \theta_n]}(E_1, \ldots, E_n)) \equiv \tau_g^k(\cup^{[\theta_1, \ldots, \theta_n]}(\tau_g^k(E_1), \ldots, \tau_g^k(E_n))) \tag{18}$$

In fact, since the Union operator keeps all the (distinct) tuples coming from each E_i expression, the k best matching tuples can be searched only among the best k of each E_i. Since g does not refer to tuples' scores, even if scores can change owing to Union, this does not influence the choice of the relevant tuples. On the other hand, if the ranking criterion is based on tuples' scores, the above rule can be applied only if Union is unweighted and the $\mathcal{FS}$ (max) semantics is used:

$$\tau^k(\cup(E_1, \ldots, E_n)) \equiv \tau^k(\cup(\tau^k(E_1), \ldots, \tau^k(E_n))) \tag{19}$$

As a negative result, we state that in no case operands of a (weighted) Join followed by a Top can be reduced by pushing down the Top (as done for Union). In fact, it is not guaranteed that the k best matching tuples of each operand finally contribute to the Join, possibly leading to the worst case, i.e. an empty set of tuples.

Considering the relationship between Top and Cut operators and the expression $\gamma_f(\tau_g^k(E))$, in the general case no manipulations are possible. On the other hand, when $f = g$, we have the following result:

$$\gamma_f(\tau_f^k(E)) \equiv \tau_f^k(\gamma_f(E)) \equiv \tau_f^k(E) \tag{20}$$

In fact, if ranking and filtering are made on the same formula f, since the k best matching tuples satisfy also this formula, inner or outer Cut's are redundant, since the filtering condition is already expressed by the ranking criterion and, as a consequence, it can be eliminated. On the other hand, the former equivalence is also obtained when both operators have their "canonical" form, that is:

$$\tau^k(\gamma_\alpha(E)) \equiv \gamma_\alpha(\tau^k(E)) \tag{21}$$

However, with regard to Cut's, elimination is not possible anymore, since the ranking rule on tuple scores does not require that they must be higher than α.

4.3 Dealing with Selection

Since the Selection operator changes tuples' scores, particular care has to be paid when reasoning about it. For instance, the following equivalence holds only for the $\mathcal{FS}$ semantics (and unweighted Union):

$$\sigma_f(\cup(E_1, \ldots, E_n)) \equiv \cup(\sigma_f(E_1), \ldots, \sigma_f(E_n)) \tag{22}$$

In fact, since in $\mathcal{FS}$ the $s_\wedge = \min$ and $s_\vee = \max$ operators are used to compute the scores of tuples after a Selection and a Union, respectively, since distributivity holds for both min and max, the order of tuple Selection, before or after a Union, is not relevant at all, since the same score is produced. On the other hand, also the same set of tuples is generated, likewise asserted by the corresponding rule of the relational algebra.

Consider now the case where $f = p_1 \wedge p_2 \wedge \ldots \wedge p_n$, with p_i over the schema of E_i, and $\Theta_f = [\theta_1, \theta_2, \ldots, \theta_n]$. The following rule shows what happens if predicates are pushed down a Join, provided no weights are used:

$$\bowtie(\sigma_{p_1}(E_1), \ldots, \sigma_{p_n}(E_n)) \equiv \sigma_f(\bowtie(E_1, \ldots, E_n)) \tag{23}$$

The above rule holds thanks to associativity of t-norms, the definitions of Join and Selection, and by observing that both expressions return the same set of tuples as it happens in RA.

4.4 Using Rules

The following example shows how the above rules can be used to transform SAME^W expressions for optimization.

Example 8. Let consider the query which returns information on the three "best" fingerprints of thumbs, with a feature vector `FP_FV` *similar to a given value* `fv` *(weight 0.4), corresponding to persons having brown hair (weight 0.6). The resulting score has to be greater than 0.5.*

This can be expressed by the following expression:

$$\gamma_{0.5}(\tau^3(\sigma_{Hair=\text{`}brown\text{'}^{0.6} \wedge FP\_FV \sim \mathtt{fv}^{0.4}}(\sigma_{FingerNo=\text{`}1\text{'}}(Faces \bowtie FingerPrints)))) \tag{24}$$

Thanks to rule (21), the outer Cut can be swapped with the Top so that rule (17) can be applied, and predicate on `Hair` and the Cut can be pushed-down as shown below. This is possible since tuples resulting from the Selection on `FingerNo` to the Join of `Faces` and `FingerPrints` are again crisp tuples.

$$\tau^3(\gamma_{0.5}(\sigma^{[0.4,0.6]}_{FP\_FV \sim \mathtt{fv}}(\gamma_{0.5}(\sigma_{Hair=\text{`}brown\text{'}}(\sigma_{FingerNo=\text{`}1\text{'}}(Faces \bowtie FingerPrints)))))) \tag{25}$$

Then, from the associativity of t-norms and the definition of Selection:

$$\tau^3(\gamma_{0.5}(\sigma^{[0.4,0.6]}_{FP\_FV \sim \mathtt{fv}}(\gamma_{0.5}(\sigma_{Hair=\text{`}brown\text{'} \wedge FingerNo=\text{`}1\text{'}}(Faces \bowtie FingerPrints))))) \tag{26}$$

Now, the application of rule (23) leads to:

$$\tau^3(\gamma_{0.5}(\sigma^{[0.4,0.6]}_{FP\_FV\sim\mathtt{fv}}(\gamma_{0.5}(\sigma_{Hair='brown'}(Faces) \bowtie \sigma_{FingerNo='1'}(FingerPrints))))) \quad (27)$$

Then, rule (10) can be applied, thus producing:

$$\tau^3(\gamma_{0.5}(\sigma^{[0.4,0.6]}_{FP\_FV\sim\mathtt{fv}}(\gamma_{0.5}(\gamma_{0.5}(\sigma_{Hair='brown'}(Faces)) \bowtie \gamma_{0.5}(\sigma_{FingerNo='1'}(FingerPrints)))))) \quad (28)$$

The Cut on `FingerPrints` can be eliminated, since the relation is crisp and the Selection has a boolean predicate, thus scores of tuples are equal to 1. Also the Cut external to the Join can be eliminated since the former Join operand has tuple scores higher than 0.5, whereas the latter one is crisp. As a consequence, due to Property 1, the resulting scores of the Join tuples are also higher than 0.5, and so the external Cut is redundant. Finally, the above expression reduces to:

$$\tau^3(\gamma_{0.5}(\sigma^{[0.4,0.6]}_{FP\_FV\sim\mathtt{fv}}(\gamma_{0.5}(\sigma_{Hair='brown'}(Faces)) \bowtie \sigma_{FingerNo='1'}(FingerPrints)))) \quad (29)$$

In conclusion, by considering the similarity predicate on fingerprints as a probably costly one, in the last expression we evaluate it only on tuples previously satisfying the other (cheap) predicates.

5 Related Work and Conclusion

Over the years, many works have been proposed on methods for representing and reasoning with "imprecise" information [Par96]. These works are only marginally relevant here, since they mostly concentrate on modeling aspects (see e.g. [RM88]) and typically ignore issues related to advanced processing of similarity queries, which are a major concern in multimedia environments. Indeed, it is a fact that similarity queries arise even if the database does not store imprecise information at all, provided "similarity operators" are defined. This is also the scenario considered by the VAGUE system [Mot88], where, however, important features are missing, such as weights, the Top operator, and fuzzy attributes. Further, problems related to query optimization are not considered in [Mot88]. Recent work by Adali et al. [ABSS98] addresses issues similar to ours, but important differences exist. First, they do not consider weights, that we have shown to introduce new interesting problems in the query optimization scenario. Second, they are mainly concentrated on problems related to the *integration* of heterogeneous "similarity measures", coming from different sources, into a common framework. Finally, several works are related to ours from the point of view of query processing and execution. In [CPZ98] complex similarity queries are evaluated through distance measures, so that execution can benefit of efficient distance-based access methods [CPZ97]. In [Fag96] a specific algorithm is proposed so as to efficiently process the so-called "Top queries", which deal with user limitation on the cardinality of query results. This argument has been an interesting subject for recent research [CK97], and a tigh connection can be found with our Top (τ) operator. Optimization techniques considered in [CK97] to "push-down" the Top basically exploit primary key-foreign key joins (as well as analysis of residual predicates) – a thing which we could embed into SAME^W by means of functional dependencies.

In this paper we have introduced a "similarity algebra with weights", called SAME^W, that generalizes relational algebra to allow the formulation of complex similarity queries over multimedia databases. SAME^W combines within a single framework several aspects relevant to multimedia queries, such as new operators (Cut and Top) useful for "range" and "best-matches" queries, weights to express user preferences, and "scores" to rank tuples. These aspects pose new challenges to a query engine, which have not been considered yet in their full generality. For instance, if the user evaluates the result of a query, thus the system should adapt to this by adjusting weights, how

does this affect execution costs? Indeed, as shown in Section 4, changing the weights will modify the numerical values, thus the processing costs, used by some operators (like the Cut) to limit the cardinality of the arguments of n-ary operators, such as Join and Union.

References

[ABSS98] S. Adali, P. Bonatti, M.L. Sapino, and V.S. Subrahmanian. A Multi-Similarity Algebra. In *Proc. of the 1998 ACM-SIGMOD Int. Conf. on Management of Data*, pp. 402–413, Seattle, WA, Jun 1998.

[CK97] M.J. Carey and D. Kossmann. On Saying "Enough Already!" in SQL. In *Proc. of the 1997 ACM SIGMOD Int. Conf. on Management of Data*, pp. 219–230, Tucson, AZ, May 1997.

[CMPT00] P. Ciaccia, D. Montesi, W. Penzo, and A. Trombetta. Imprecision and User Preferences in Multimedia Queries: A Generic Algebraic Approach. In *Proc. of the Int. Symp. on Foundations of Information and Knowledge Systems (FoIKS 2000)*, vol. 1762 of *LNCS*, pp. 50–71, Burg, Germany, Feb 2000, Springer-Verlag.

[CPZ97] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proc. of the 23rd VLDB Int. Conf.*, pp. 426–435, Athens, Greece, Aug 1997.

[CPZ98] P. Ciaccia, M. Patella, and P. Zezula. Processing complex similarity queries with distance-based access methods. In *Proc. of the 6th Int. Conf. on Extending Database Technology (EDBT'98)*, vol. 1377 of *LNCS*, pp. 9–23. Springer-Verlag, Valencia, Spain, Mar 1998.

[Fag96] R. Fagin. Combining Fuzzy Information from Multiple Systems. In *Proc. of the 15th ACM Symp. on Principles of Database Systems (PODS'96)*, pp. 216–226, Montreal, Canada, Jun 1996.

[FSN+95] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by Image and Video Content: The QBIC System. *IEEE Computer*, 28(9):23–32, Sep 1995.

[FW97] R. Fagin and E.L. Wimmers. Incorporating User Preferences in Multimedia Queries. In *Proc. of the 6th ICDT Int. Conf.*, pp. 247–261, Delphi, Greece, Jan 1997.

[KY95] G.J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic*. Prentice Hall PTR, 1995.

[LMM97] A. Lumini, D. Maio, and D. Maltoni. Continuous versus Exclusive Classification for Fingerprint Retrieval. *Pattern Recognition Letters*, 18:1027–1034, 1997.

[Mot88] A. Motro. VAGUE: A User Interface to Relational Databases that Permits Vague Queries. *ACM Trans. on Office Information Systems*, 6(3):187–214, Jul 1988.

[MT99] D. Montesi and A. Trombetta. Similarity Search through Fuzzy Relational Algebra. In *Proc. of the 1st Int. Workshop on Similarity Search (IWOSS'99)*, Florence, Italy, Sep 1999.

[NRT99] S. Nepal, M.V. Ramakrishna, and J.A. Thom. A Fuzzy Object Language (FOQL) for Image Databases. In *Proc. of the 6th Int. Conf. on Database Systems for Advanced Applications (DASFAA'99)*, pp. 117–124, Hsinchu, Taiwan, Apr 1999.

[Par96] Simon Parsons. Current Approaches to Handling Imperfect Information in Data and Knowledge Bases. *IEEE Trans. on Knowledge and Data Engineering*, 8(3):353–372, 1996.

[RM88] K. Raju and A. Majumdar. Fuzzy Functional Dependencies and Lossless Join Decomposition of Fuzzy Relational Database Systems. *ACM Trans. on Database Systems*, 13(32):129–166, Jun 1988.

[SS98] A. Soffer and H. Samet. Integrating Symbolic Images into a Multimedia Database System using Classification and Abstraction Approaches. *The VLDB Journal*, 7(4):253–274, 1998.