Principles of Information Filtering in Metric Spaces

Paolo Ciaccia and Marco Patella
DEIS, Università di Bologna - Italy
{paolo.ciaccia,marco.patella}@unibo.it

***Abstract*—The traditional problem of similarity search requires to find, within a set of points, those that are closer to a query point q, according to a distance function d. In this paper we introduce the novel problem of metric filtering: in this scenario, each data point x_i possesses its own distance function d_i and the task is to find those points that are close enough, according to d_i, to a query point q. This minor difference in the problem formulation introduces a series of challenges from the point of view of efficient evaluation. We provide basic definitions and alternative pivot-based resolution strategies, presenting results from a preliminary experimentation that show how the proposed solutions are indeed effective in reducing evaluation costs.**

I. Introduction

Metric spaces are a powerful framework to describe a variety of search problems that commonly arise in several application domains [1], [2]. For instance, the metric space model has been successfully adopted in content-based retrieval of multimedia objects, in genomic and biomolecular databases, as well as for string/text databases.

Although the metric space model is general enough to satisfy the requirements of many contexts, it lacks a relevant feature, which is becoming more and more important for novel database applications, that is, the possibility of accommodating *user preferences* in the specification of the distance function that determines how much two objects can be considered to be "similar" to each other. This leads to consider an "enlarged" metric scenario, allowing "personalized views" of the space. The simplest case to be examined is when each query q carries its own personal metric, according to which objects are to be ranked: this *user-defined queries* scenario can be dealt with thanks to recent solutions applicable to generic metric spaces whenever personal metrics all derive from a "default" (natural) metric [3].

The more general case, where each point of the space is associated to its personal metric, represents a generalization of information filtering [4], where a profile is stored for each user of the system and, each time a new information becomes available, it is forwarded only to those users whose profile is sufficiently similar to it (according to the user distance criterion). Clearly, this extension provides the maximum of freedom, since no restrictions are put on the personal metrics. In the following we describe how the model of metric spaces could be extended to the case where each object "sees" its own metric world. At the same time, we provide some hints that could be exploited by search algorithms to guarantee good performance levels.

II. The Problem

Given a domain $\mathcal{U}$ of objects ($\mathcal{U}$ is the "universe"), a metric space over $\mathcal{U}$ is a pair $\mathcal{M} = (\mathcal{U}, d)$, where d is a distance function that satisfies the metric postulates $\forall x, y, z \in \mathcal{U}$:

$$\begin{aligned} &d(x,y) \geq 0 && \text{(non-negativity)}\\ &d(x,y) = 0 \iff x = y && \text{(indiscernibility)}\\ &d(x,y) = d(y,x) && \text{(symmetry)}\\ &d(x,z) \leq d(x,y) + d(y,z) && \text{(triangle inequality)} \end{aligned}$$

Consider now a finite subset X of $\mathcal{U}$, called a *dataset*, and a query point $q \in \mathcal{U}$: each point $x_i \in X$ is associated to a *personal metric* d_i and a *personal radius* r_i. In information filtering terms, the triple (x_i, d_i, r_i) represents the *profile* of the ith user, whereas q is a new *item* (product). The system's objective is then to deliver q only to those users whose profiles "match" the new item q, i.e., for which $d_i(x_i, q) \leq r_i$.[1] We call such set of points the *query result*, with a slight abuse of terminology.

This generalizes the Multi-Modal approach presented in [4], that represents information as spatial points, according to the vector-space model [5]: users are modelled as (sets of) points in the same space and information is forwarded only to users whose points are close enough (according to a single distance d) to the query point, thus $d_i \equiv d, \forall i$. The focus of [4] is on the efficient updating of user profiles: such results could be easily extended for updating personalized metrics, e.g., by exploiting results on relevance feedback [6].

It has to be remarked that the scenario we consider here is rather different from the case of user-defined queries, where a single distance d_q is considered at query time: results in [3] indeed show how metric access methods can be modified so as to efficiently process user-defined queries, the key idea being to exploit a bound on the ratio between d_q and a default distance d_0 to limit the search space. Indeed, if $d_0(q,x)/d_q(q,x) \leq s, \forall x$, then knowing that $d_0(q,x) > s \cdot r$ is sufficient to deduce that $d_q(q,x) > r$. The case of "metric filtering" appears to be much more challenging to deal with,

[1] The case where each user is associated with multiple profiles can be straightforwardly defined.

since now we have, at each instant, $|X|$ personal metrics in play (one for each element of the dataset). Indeed, if no reasonable assumption is made on the d_i metrics, it is not possible to exploit the triangle inequality to speedup the search process. To see why this is the case, consider that all metric access methods rely on a *pivot-based comparison* to avoid computing some distances. This is clearly true for pivot-based methods (starting from AESA [7]), but also holds for partition-based methods (such as the M-tree [8]). For instance, the M-tree relies on information on the covering radius of a node to decide whether that node has to be accessed or not. Clearly, this information is useful as long as the covering radius is an upper-bound of the distances of objects in that node from the routing object (center) of the node, so that the triangle inequality (written in the "difference" form) can provide a lower-bound on the distances from objects in the node to the query point. Since similar issues are to be considered for both pivot-based and partition-based methods, in the following we consider the simplest case of pivot-based methods, being understood that, with suitable modifications, our observations apply to partition-based methods as well.

III. Pivot-based Methods for Metric Filtering

With pivot-based methods the basic idea is to start by choosing (at random or possibly using some "smart" criterion [9]) a set of m *pivots* $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$, $\mathcal{P} \subseteq X$, and then measure and store all the distances between such pivots and the elements of the dataset X.[2] When the metric d is in use, the inequality $|d(q, p_j) - d(p_j, x_i)| > r$ is enough to conclude that $d(q, x_i) > r$, thus leading to exclude x_i from the result without the need of computing $d(q, x_i)$.

Assume now a "metric filtering" scenario, where both p_j and x_i use their own metrics. Using the inequality $|d_j(q, p_j) - d_j(p_j, x_i)| > r_i$ one can only deduce that $d_j(q, x_i) > r_i$, but, without further assumptions, there is no way to conclude that also $d_i(q, x_i) > r_i$ holds.

Also observe that the inequality

$$|d_i(q, p_j) - d_i(p_j, x_i)| > r_i, \tag{1}$$

which would be the standard way to infer that $d_i(q, x_i) > r_i$, cannot be used without giving up the very role of pivots. Indeed, using $d_i(q, p_j)$ (rather than $d_j(q, p_j)$) to measure the distance between q and p_j would require to compute the distance between q and p_j for each x_i, thus making pivots completely useless! Figure 1 makes explicit the reference scenario to consider.

We are interested in bounding from below[3] the value $d_i(q, x_i)$, under the following constraints:

[2]In principle, pivots do not need to be elements of the dataset X.

[3]Bounding from above the value of $d_i(q, x_i)$ can lead to conclude that q is relevant for x_i if the bound is not larger than r_i; we never use such inequality in the following.

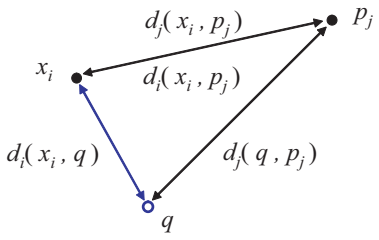

Figure 1. The basic distances

1) the d_i metric cannot be used to compare pivot p_j with query q (the basic choice is to use d_j to this purpose);
2) either $d_i(x_i, p_j)$ or $d_j(x_i, p_j)$ (or both) can be pre-computed and stored.

Because of the first constraint, one must at least assume that metrics d_i and d_j are somewhat related, so that measuring $d_j(q, p_j)$ can provide some information on $d_i(q, p_j)$ and, in turn, on $d_i(q, x_i)$. To this end we rely on the well-known lower-bounding relationships between distance functions. In particular we consider *scaled* lower bounds, as introduced in [3].

Definition 1. *Let d_i and d_j be the metrics used by point x_i and pivot p_j, respectively. We say that d_i is a* scaled lower bound *of d_j if and only if it exists a real value $s_{i,j} > 0$ such that $\forall x, y \in \mathcal{U}$ it is:*

$$d_i(x, y) \leq s_{i,j} \cdot d_j(x, y), \tag{2}$$

which can also be concisely written as $d_i \preceq s_{i,j} \cdot d_j$. The minimum value of $s_{i,j}$ for which the inequality holds is called the (optimal) scaling factor *of d_j with respect to d_i. In the following, only optimal scaling factors are used.*

In a similar way, we have that the pivot metric is a scaled lower bound of d_i if and only if there exists $s_{j,i} > 0$ such that:

$$d_j(x, y) \leq s_{j,i} \cdot d_i(x, y), \tag{3}$$

also written as $d_j \preceq s_{j,i} \cdot d_i$.

Basic properties of scaling factors are proved in the following lemma.

Lemma 1. *For any two metrics d_i and d_j, if both scaling factors $s_{i,j}$ and $s_{j,i}$ exist, then it is*

$$s_{i,j} \cdot s_{j,i} \geq 1. \tag{4}$$

with equality being achieved if and only if d_j is a rescaling of d_i, that is, $d_j(x, y) = s_{j,i} \cdot d_i(x, y), \forall x, y \in \mathcal{U}$ (in this case, the two corresponding metric spaces are isometric).

Proof: From the definition of scaling factors it is $d_i(x, y) \leq s_{i,j} \cdot d_j(x, y) \leq s_{i,j} \cdot s_{j,i} \cdot d_i(x, y)$, from which, after dividing by $d_i(x, y)$, it is derived $s_{i,j} \cdot s_{j,i} \geq 1$.

When $d_j(x,y) = s_{j,i} \cdot d_i(x,y)$, it is plain to see that $s_{i,j} = 1/s_{j,i}$. Going the other side, if equality holds in (4) it is $d_j \preceq s_{j,i} \cdot d_i = d_i/s_{i,j} \preceq d_j$, from which the result follows. ∎

Example 1. *Let d_i and d_j be two weighted Euclidean metrics on $\mathbb{R}^2 \times \mathbb{R}^2$, respectively defined as:*

$$d_i(x,y) = \sqrt{(x[1]-y[1])^2 + 2 \cdot (x[2]-y[2])^2}$$
$$d_j(x,y) = \sqrt{3 \cdot (x[1]-y[1])^2 + (x[2]-y[2])^2}$$

In the case of weighted Euclidean distances, it is not difficult to show that the scaling factor $s_{i,j}$ is given by the square root of the maximum ratio of corresponding weights $w[k]$ of the two metrics, that is, $s_{i,j} = \max_k \sqrt{w_i[k]/w_j[k]}$ (analogously for the derivation of $s_{j,i}$). For our sample distances it is therefore derived that $s_{i,j} = \sqrt{2}$ and $s_{j,i} = \sqrt{3}$.

Based on the assumption that at least one of the scaling factors $s_{i,j}$ and $s_{j,i}$ exists, in the following we list the basic alternatives to generalize pivot-based methods so as to deal with the metric filtering problem.

A. Computing Bounds in Pivot Spaces

The first alternative we consider assumes that for each point x_i and pivot p_j the distance $d_j(p_j,x_i)$ is precomputed and stored. Since the distance between p_j and q is necessarily measured using d_j, computation is indeed done only in "pivot spaces". In this case there are two basic strategies that can be used to prune point x_i.

Triangle inequality in pivot space. Assume that $d_j \preceq s_{j,i} \cdot d_i$. Then:

$$\frac{|d_j(q,p_j) - d_j(p_j,x_i)|}{s_{j,i}} > r_i \tag{5}$$

is a sufficient condition to guarantee that x_i can be pruned. Indeed, from (3) and (5) it is derived $d_i(q,x_i) \geq d_j(q,x_i)/s_{j,i} \geq |d_j(q,p_j) - d_j(p_j,x_i)|/s_{j,i} > r_i$.

Using both scaling factors. When both scaling factors are defined, the following inequality could also be used to prune x_i:

$$\frac{d_j(q,p_j)}{s_{j,i}} - s_{i,j} \cdot d_j(p_j,x_i) > r_i \tag{6}$$

Correctness is easily proved, since it is $d_i(q,x_i) \geq d_i(q,p_j) - d_i(p_j,x_i) \geq d_j(q,p_j)/s_{j,i} - d_i(p_j,x_i) \geq d_j(q,p_j)/s_{j,i} - s_{i,j} \cdot d_j(p_j,x_i) > r_i$. However, let us consider the "positive form" of (5), that is

$$\frac{d_j(q,p_j)}{s_{j,i}} - \frac{d_j(p_j,x_i)}{s_{j,i}} > r_i. \tag{7}$$

We can exploit Lemma 1 to immediately conclude that $s_{i,j} \cdot d_j(p_j,x_i) \geq d_j(p_j,x_i)/s_{j,i}$, since $s_{i,j} \geq 1/s_{j,i}$. This means that (7) will always yield a better (higher) lower bound to $d_i(q,x_i)$ than (6). In logical terms this is also to say that (6) implies (7), written (6) $\implies$ (7).[4]

[4]Note that a lower-bounding inequality I_{useful} is useful as long as there is no other inequality I_{other} such that $I_{\text{useful}} \implies I_{\text{other}}$.

Similar arguments can be applied if one exchanges the roles of $d_j(q,p_j)$ and $d_j(p_j,x_i)$, thus using the inequality

$$\frac{d_j(p_j,x_i)}{s_{j,i}} - s_{i,j} \cdot d_j(q,p_j) > r_i, \tag{8}$$

and compares it against the "negative form" of (5), that is

$$\frac{d_j(p_j,x_i)}{s_{j,i}} - \frac{d_j(q,p_j)}{s_{j,i}} > r_i. \tag{9}$$

Again, due to Lemma 1 it is $s_{i,j} \cdot d_j(q,p_j) \geq d_j(q,p_j)/s_{j,i}$, thus (8) $\implies$ (9).

In conclusion, we see that the strategy that uses both scaling factors is never convenient.

B. Computing Bounds in Point Spaces

In the above-described approaches precomputed distances between points and pivots are measured using the pivots' metrics. The other alternative to consider, aiming to possibly derive more accurate bounds, is to make use of $d_i(p_j,x_i)$ in place of $d_j(p_j,x_i)$ (remind that the distance between q and p_j is necessarily measured using d_j).

Large pivot-point distance. Consider (1): since $d_i(q,p_j) \leq s_{i,j} \cdot d_j(q,p_j)$, we can use the inequality

$$d_i(p_j,x_i) - s_{i,j} \cdot d_j(q,p_j) > r_i \tag{10}$$

to prune point x_i.

Small pivot-point distance. When point x_i is "close" to pivot p_j the triangle inequality in point space can also be used to prove that

$$\frac{d_j(q,p_j)}{s_{j,i}} - d_i(p_j,x_i) > r_i \tag{11}$$

is a valid pruning condition. This follows from the inequalities $d_i(q,x_i) \geq d_i(q,p_j) - d_i(p_j,x_i) \geq d_j(q,p_j)/s_{j,i} - d_i(p_j,x_i)$.

However this approach is not effective at all, when compared to the (higher) lower bound provided by (7), since (11) $\implies$ (7). This easily follows from the observation that $d_i(p_j,x_i) \geq d_j(p_j,x_i)/s_{j,i}$.

Before proceeding it is useful to summarize the results obtained so far. These are as follows:

1) The scaling factor $s_{i,j}$, as defined in (2), is not useful at all to bound the search space if we are working only in "pivot space", whereas it can be exploited through (10) if we also work in "point space".
2) Among the several possible conditions that can be tested to prune x_i, only three of them are non-redundant, namely:

$$\frac{d_j(q,p_j)}{s_{j,i}} - \frac{d_j(p_j,x_i)}{s_{j,i}} > r_i, \tag{7}$$
$$\frac{d_j(p_j,x_i)}{s_{j,i}} - \frac{d_j(q,p_j)}{s_{j,i}} > r_i, \tag{9}$$
$$d_i(p_j,x_i) - s_{i,j} \cdot d_j(q,p_j) > r_i. \tag{10}$$

The strategy that uses all above inequalities (which we call Δ-both) has to pay for an increased storage overhead and preprocessing time, since both $d_j(p_j, x_i)$ and $d_i(p_j, x_i)$ have to be used. We will experimentally investigate under which conditions, depending on the distance(s) between p_j and x_i, it is convenient to trade off space for bounds accuracy by storing only one of these two distances. In this case, we can either use tests (7) and (9) (Δ-pivot strategy) or the combination of (10) and the otherwise redundant (11) (Δ-point strategy).

IV. Beyond Correctness: The Symmetric Scaling Factor

In order to characterize not only the correctness but also the performance of metric filtering, it is important to properly understand how the use of a specific lower-bounding inequality (among those introduced in the previous section) can impact search costs. To this end, in this section we provide basic results aiming to shed more light on the role that scaling factors play from a performance point of view. We start with the following preliminary observations:

1) Just considering the scaling factor $s_{j,i}$ (or $s_{i,j}$) says nothing about "how well" d_j approximates d_i. This is also to say that the magnitude of $s_{j,i}$ ($s_{i,j}$) does not matter at all.
2) As demonstrated by Lemma 4, when $s_{i,j} \cdot s_{j,i} = 1$ we have an "ideal" situation in that d_j, albeit different from d_i, equals d_i up to a scaling factor.
3) The bound(s) provided by approximate distances can be even higher (thus better) than the bound(s) obtainable from (1).

The last point is quite surprising, since it tells us that there are cases where scaled bounds are more effective than "natural" bounds, as the following example demonstrates.

Example 2. *Let d_i and d_j be the two weighted Euclidean metrics defined in Example 1. Let $x_i = (7, 5)$, $p_j = (7, 8)$, and $q = (3, 5)$, from which it is derived $d_i(q, x_i) = \sqrt{(3-7)^2 + 2 \cdot (5-5)^2} = 4$. The "natural" bound yielded by (1) would be*

$$d_i(q, x_i) \geq |d_i(q, p_j) - d_i(p_j, x_i)| = |\sqrt{34} - \sqrt{18}| \approx 1.588.$$

On the other hand, the bound computed using (7) is

$$d_i(q, x_i) \geq \frac{d_j(q, p_j) - d_j(p_j, x_i)}{s_{j,i}} = \frac{\sqrt{57} - \sqrt{9}}{\sqrt{3}} \approx 2.626.$$

Assuming a personal radius $r_i = 2$, it is evident that using the pivot's metric could lead to discard x_i, whereas this would not be the case if pruning were based on the classical triangle inequality.

Based on above observations, we introduce the *symmetric scaling factor* (SSF for short) of d_i and d_j, defined as $S(i, j) = s_{i,j} \cdot s_{j,i}$, as a synthetic yet effective way to characterize the relationship between d_i and d_j metrics. SSF can be given a simple interpretation, which is formalized by the next lemma and illustrated in the following example.

Lemma 2. *It is*

$$\sup_{\mathcal{U}} \{d_i(x, y) | d_j(x, y) = s_{j,i}\} \leq S(i, j),$$
$$\sup_{\mathcal{U}} \{d_j(x, y) | d_i(x, y) = s_{i,j}\} \leq S(i, j).$$

Proof: Immediate from the definition of scaling factors and of SSF. ■

Example 3. *Let d_i and d_j be the two weighted Euclidean metrics defined in Example 1. Since $s_{i,j} = \sqrt{2}$ and $s_{j,i} = \sqrt{3}$, it is $S(i, j) = \sqrt{6}$. Consider now the set of points $Y_i = \{y \in \mathcal{U} : d_i(x, y) \leq 1\}$. Since $s_{j,i} = \sqrt{3}$, we have that for all $y \in Y_i$ it is also guaranteed $d_j(x, y) \leq \sqrt{3}$, because $d_j(x, y)/\sqrt{3} \leq 1$ holds. In geometrical terms, the Y_i ellipsoid centered in x with equation $d_i(x, y) \leq 1$ is contained in the co-centric ellipsoid Y_j with equation $d_j(x, y) \leq \sqrt{3}$ (see Figure 2).*

Consider now a point on the surface of Y_j and measure its distance from x using *d_i. According to Lemma 2, this distance cannot exceed the value of the* SSF*, $S(i, j) = \sqrt{6}$.[5] Intuitively, the higher the value of $S(i, j)$, the worse Y_j (d_j) approximates Y_i (d_i).*

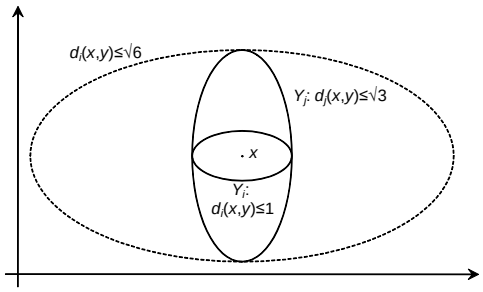

Figure 2. The effect of scaling factors and of SSF.

We can generalize the observations in the above example as follows. When d_j is scaled so as to become a lower bound of d_i ($d_j/s_{j,i} \preceq d_i$), if $S(i, j) > 1$ then such scaling leads to define the (normalized) non-empty region:

$$R(x) = \{y | d_i(x, y) - d_j(x, y)/s_{j,i} > 0, d_j(x, y) = s_{j,i}\}$$
$$= \{y | d_i(x, y) > 1, d_j(x, y) = s_{j,i}\}$$

Within this region we have all points for which the bound provided by $d_j(x, y)/s_{j,i}$ is not accurate, and Lemma 2 simply says that

$$\sup \{d_i(x, y) | y \in R(x)\} \leq S(i, j).$$

[5] In the example the $\sqrt{6}$ bound is achieved if one takes the points $y \equiv (x[1], x[2] \pm \sqrt{3})$.

Therefore, the SSF of d_i and d_j tells us *how much, in the worst case, we "relax" d_i by approximating it with d_j*.

It has to be observed that the result does not depend on the specific reference point x and that it also holds when the roles of d_i and d_j are reversed (since $S(i,j) = S(j,i)$).

The basic properties of SSFs are as follows:

$$\begin{aligned} &S(i,j) \geq 1, \\ &S(i,j) = 1 \iff d_i \text{ is a rescaling of } d_j, \\ &S(i,j) = S(j,i), \\ &S(i,k) \leq S(i,j) \cdot S(j,k), \end{aligned}$$

where the last inequality easily derives from the definition of scaling factors. For any two metric distances d_i and d_j, $\log S(i,j)$ is therefore a pseudo-metric among them.

A. Analyzing Lower-Bounding Inequalities

We come now to consider things from a more pragmatical point of view, that is: what is the effect on performance when we bound from below the value of $d_i(q, x_i)$ using one of the inequalities introduced in Section III? In particular we consider, for each lower-bounding expression, its relationship with the "natural" bound provided by (1).

A. Consider the expression in (7). From the definition of scaling factor and of SSF it is derived:

$$\begin{aligned} \frac{d_i(q,p_j)}{S(i,j)} - d_i(p_j,x_i) &\leq \frac{d_j(q,p_j)}{s_{j,i}} - \frac{d_j(p_j,x_i)}{s_{j,i}} \\ &\leq d_i(q,p_j) - \frac{d_i(p_j,x_i)}{S(i,j)}. \end{aligned} \tag{12}$$

B. For the expression in (9) it is similarly derived:

$$\begin{aligned} \frac{d_i(p_j,x_i)}{S(i,j)} - d_i(q,p_j) &\leq \frac{d_j(p_j,x_i)}{s_{j,i}} - \frac{d_j(q,p_j)}{s_{j,i}} \\ &\leq d_i(p_j,x_i) - \frac{d_i(q,p_j)}{S(i,j)}. \end{aligned} \tag{13}$$

C. The expression in (10) admits the following bounds in point space:

$$\begin{aligned} &d_i(p_j,x_i) - S(i,j) \cdot d_i(q,p_j) \leq \\ &\quad d_i(p_j,x_i) - s_{i,j} \cdot d_j(q,p_j) \leq d_i(p_j,x_i) - d_i(q,p_j). \end{aligned} \tag{14}$$

D. Finally, for the expression in (11) it is:

$$\begin{aligned} \frac{d_i(q,p_j)}{S(i,j)} - d_i(p_j,x_i) &\leq \frac{d_j(q,p_j)}{s_{j,i}} - d_i(p_j,x_i) \\ &\leq d_i(q,p_j) - d_i(p_j,x_i). \end{aligned} \tag{15}$$

Comparing above limit cases with the triangle inequality in point space, it is clear that:

- Only cases A and B, which only work in pivot space, can provide better lower bounds than the triangle inequality in point space. On the other hand, in the most favorable situation C and D can only provide a bound equal to the one obtainable from the triangle inequality in point space.
- Case C has the advantage, with respect to case B, of enjoying a most favorable worst case, since the lower limit of C is $S(i,j)$ times higher than that of B. In other terms, considering both extremes of B and C, we can say that B is "more aggressive" than C.

V. Experimental Evaluation

We implemented the three pruning strategies proposed in Section III and tested them over some synthetic datasets. The use of synthetic, rather than real, datasets is motivated by the fact that in our experiments we need not only a set of data points, but also a corresponding distance for each point. We therefore generated three different 3D datasets in the $[0,1]^3$ cube, using a weighted Euclidean distance; data (and query) points and distance weights were produced as follows:

- **uni** Points and weights are uniformly distributed in the unit hypercube.
- **clust** Points follow a Gaussian distribution (with $\sigma^2 = 0.1$) around 5 centers that are uniformly distributed in the unit hypercube; weights are also uniformly distributed.
- **rw** (random walk) A single point is randomly chosen in the unit hypercube; following points are created by slightly perturbing (0.01 of maximum amount) the coordinates of the previous point; weights are created in a similar fashion.

While the first two datasets are commonly used as yardsticks for measuring performances of spatial/metric structures, the latter has been considered so as to investigate a situation where close objects have similar distances, which appears to be a reasonable assumption (see Figure 3). Finally, for each dataset, the personal radii, r_i, were chosen so that each point selects around 3% of incoming queries.

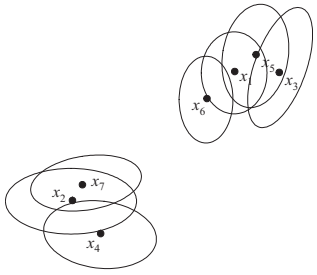

Figure 3. In the **rw** dataset, objects close in the space adopt similar metrics.

We used a number of pivots in the range $[25, 150]$: pivots were chosen so as to maximize the distance among them, according to the *outliers selection* technique [9]. We are mainly interested in evaluating the performance of the proposed strategies in terms of two basic objective measures [1]:

- **Internal complexity**: This measure accounts for the effort (in terms of distance computations) needed by the index structure to compute the set of candidate results,

i.e., those objects that *may* be part of the result, but need to be exactly checked using their respective distance functions. In our case, the internal complexity always coincides with the number of pivots used.

- **External complexity**: This is the cost needed for filtering the set of candidate results, i.e., the number of distances that have to be computed to check whether candidate objects belong to the result or are just false positives.

The alternatives that will be contrasted are listed in Table I (the Δ strategy, being based on (1), cannot be used with pivots and is only included as a reference yardstick for external complexity).

Strategy name	Used inequalities
Δ	(1)
Δ-pivot	(7) and (9)
Δ-point	(10) and (11)
Δ-both	(7), (9), and (10)

Table I
PRUNING STRATEGIES USED IN THE EXPERIMENTS.

Graphs in Figure 4 and 5 respectively show the external and the total (internal+external) complexity of the three filtering strategies; measures are normalized to the total number of points (30K for all 3 datasets) so as to show the speed-up over a simple sequential scan. As expected, the Δ-both strategy always attains the best performance, in many cases very close to the Δ strategy. Comparing the Δ-pivot and the Δ-point strategies, the latter strategy is rarely better than the former (as expected, since (11) is never more effective than (7)). Remarkably, all strategies attain very good performance on all considered datasets, never requiring to test more than 10% of the data points.

As already observed in Section IV-A, inequalities in pivot spaces (cases A and B) can provide bounds that are better than "natural" ones obtained by way of (1). In our experiments, this happened in 40% of the cases for the **uni** dataset, in 35% for **clust**, and in 20% for **rw**.

The optimal number of pivots ($\overline{m}$) can be observed in Figure 5 and ranges between 100 and 150 for the most efficient Δ-both strategy. Clearly, the dataset cardinality n has an influence on the value of $\overline{m}$, with $\overline{m}$ increasing for increasing values of n [9]. This is confirmed by graphs in Figure 6, where we plot the total complexity of the Δ-both strategy when both m and n are varied: for low values of n, the optimal number of pivots is low, while increasing the number of data objects (up to 100K for all the datasets) leads to increasing optimal values.

A. Optimizing Performances

In this section we investigate two optimizations aiming to reduce the time complexity of the pivot-based techniques proposed for metric filtering. In particular, we consider how the order according to which pivots are accessed for each data point can be modified so as to discard non-candidate objects as early as possible, i.e., visiting the least number of pivots. Clearly, this has no effect on the number of distances to be computed to assess whether a new object is relevant for a data object or not, but might reduce the number of comparisons for each data object.

The first heuristics we propose is to order pivots according to their distance to the query point, i.e., for increasing values of $d_j(q, p_j)$. This is based on the assumption that a pivot close to q is likely to have a higher pruning power with respect to a pivot which is far from q. The cost for this optimization is rather low, since it does not require any additional information and its complexity is $O(m \log m)$.

Our second strategy exploits the distribution of distances [10] for each of the m pivots to estimate the pruning probability of each pivot. Pivots are then sorted for decreasing values of such probability, so that pivots that are most likely to prune a data point are accessed first. To show how the pruning probability of a pivot can be computed, let us consider strategies working in pivot spaces, i.e., Δ-pivot and Δ-both, and a data point x_i. From (7) and (9), it follows that x_i cannot be pruned by p_j iff $d_j(p_j, x_i) \in [d_j(q, p_j) - s_{j,i} \cdot r_i, d_j(q, p_j) + s_{j,i} \cdot r_i]$. If $F_j(t)$ is the distance distribution for pivot p_j, i.e., $F_j(t) = Prob\{d_j(p_j, x) \leq t\}, x \in \mathcal{U}$, and $f_j(t)$ is its corresponding density function, $f_j(t) = dF_j(t)/dt$ (see Figure 7), then the probability $PP_{i,j}$ that x_i is pruned by p_j is given by:

$$\begin{aligned} PP_{i,j} &= 1 - (F_j(d_j(q, p_j) - s_{j,i} r_i) - F_j(d_j(q, p_j) + s_{j,i} r_i)) \\ &= 1 - \int_{d_j(q,p_j) - s_{j,i} \cdot r_i}^{d_j(q,p_j) + s_{j,i} \cdot r_i} f_j(t)\, dt \end{aligned} \tag{16}$$

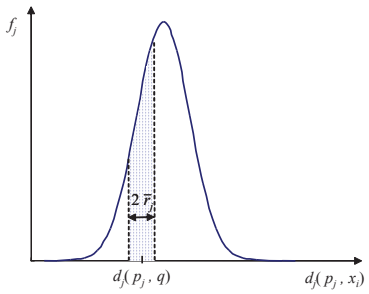

Figure 7. How the pruning power of a pivot p_j can be estimated using the density distribution of distances.

Ordering pivots for decreasing values of $PP_{i,j}$ would require computing (16) and sorting pivots for each point x_i in the dataset. A simple heuristics that allows to keep manageable costs is to sort pivots only once, by using an

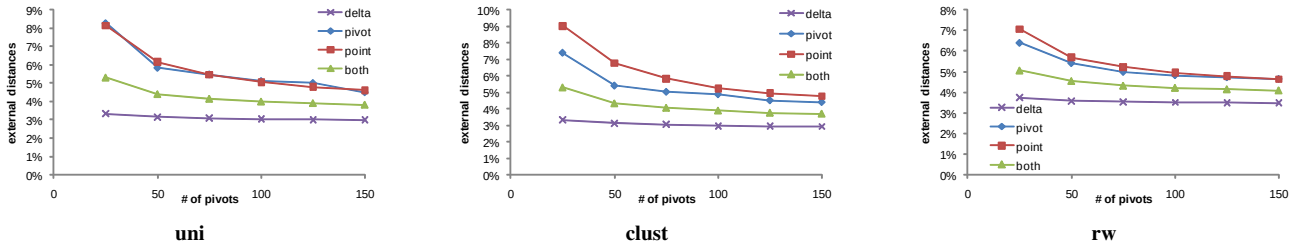

Figure 4. External complexity for the three datasets: varying number of pivots, different pruning strategies.

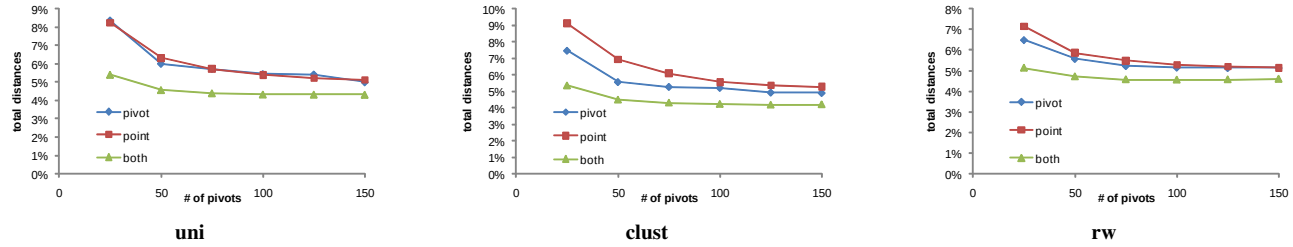

Figure 5. Total complexity for the three datasets: varying number of pivots, different pruning strategies.

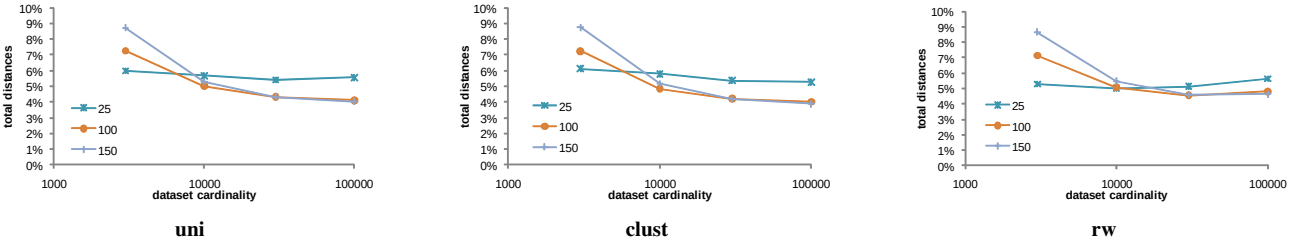

Figure 6. Total complexity for the three datasets: varying data cardinality, different number of pivots, Δ-both pruning strategy.

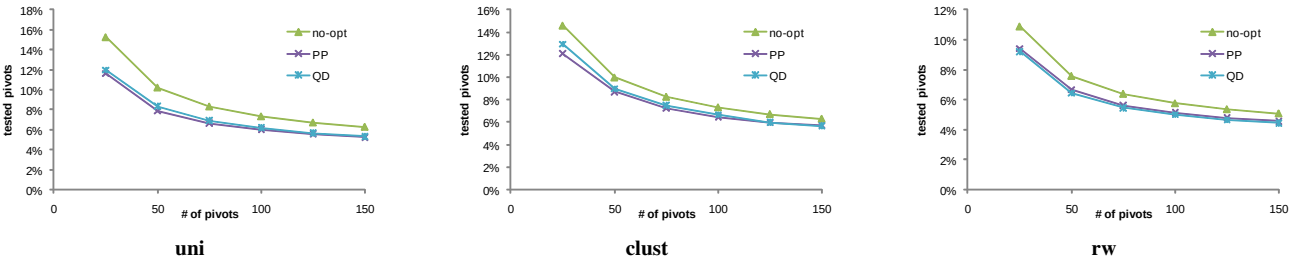

Figure 8. Fraction of tested pivots for the three datasets: varying number of pivots, Δ-both pruning strategy, different optimizations.

average effective radius $\overline{r}_j$ for each pivot, computed as

$$\overline{r}_j = \frac{1}{n}\sum_{i=1}^{n} s_{j,i} \cdot r_i \tag{17}$$

Then, substituting (17) into (16), we obtain the pruning power of pivot p_j, PP_j, as

$$\begin{aligned} PP_j &= 1 - (F_j(d_j(q,p_j) - \overline{r}_j) - F_j(d_j(q,p_j) + \overline{r}_j)) \\ &= 1 - \int_{d_j(q,p_j)-\overline{r}_j}^{d_j(q,p_j)+\overline{r}_j} f_j(t)\, dt \end{aligned}$$

Figure 8 shows the effect of the two optimizations on the Δ-both strategy for all the considered datasets (with 3K data points), computed as the fraction of pivots that have to be checked for each non-candidate point (candidate points necessarily test 100% of the pivots). Clearly, having more pivots leads to a better utilization of them, i.e., increase in the total number of comparisons is sublinear in the number of pivots. Moreover, we see that the optimization based on the distribution of distances (denoted PP in the graphs) is the one that attains the lowest number of comparisons (up to a 23.9% saving over the non-optimized Δ-both strategy for the **uni** dataset); only for the **rw** dataset the optimization based on the query distance (QD in the graphs) obtains better performance. This, however, was expected since in

the **rw** dataset close objects have also similar distances, thus favoring the QD strategy. On the other hand, the **rw** dataset is the one where the non-optimized strategy achieves the better performance, thus the saving obtainable by optimizations is limited with respect to other scenarios, reaching only 16% when using the QD optimization with 25 pivots.

VI. Conclusions

We have introduced the novel problem of metric filtering, whose main difference from the traditional problem of searching in metric spaces is the fact that every data point carries its own personal distance to measure dissimilarity with queries. To speed up query resolution over the simple sequential scan of the data points, we presented three different strategies to be applied to pivot-based access methods. Preliminary experimental results show that the proposed strategies are effective in reducing query processing costs: a more thorough study of performance is however needed. We believe that our results will be helpful in several cases of recommendation systems, where user profiles are used to suggest new items to users, according to a similarity criterion that takes into account the items each user has visited/bought in the past. Along this direction, another issue that is interesting to analyze is whether further optimizations could be devised in the case where multiple queries are submitted in a batch, which is the most common case in information filtering.

References

[1] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín, "Proximity searching in metric spaces," *ACM Computing Surveys*, vol. 33, no. 3, pp. 273–321, Sep. 2001.

[2] E. Chávez and G. Navarro, Eds., *Proceedings of the First International Workshop on Similarity Search and Applications (SISAP 2008)*. Cancún, Mexico: IEEE Computer Society, Apr. 2008.

[3] P. Ciaccia and M. Patella, "Searching in metric spaces with user-defined and approximate distances," *ACM Transactions on Database Systems*, vol. 27, no. 4, pp. 398–437, Dec. 2002.

[4] U. Çetintemel, M. J. Franklin, and C. L. Giles, "Self-adaptive user profiles for large-scale data delivery," in *Proceedings of the 16th International Conference on Data Engineering (ICDE 2000)*. San Diego, CA: IEEE Computer Society, Mar. 2000, pp. 622–633.

[5] G. Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Reading, MA: Addison-Wesley, 1989.

[6] I. Bartolini, P. Ciaccia, and F. Waas, "FeedbackBypass: A new approach to interactive similarity query processing," in *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 2001)*. Rome, Italy: Morgan Kaufmann, Sep. 2001, pp. 201–210.

[7] E. Vidal, "An algorithm for finding nearest neighbours in (approximately) constant average time," *Pattern Recognition Letters*, vol. 4, no. 3, pp. 145–157, 1986.

[8] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*. Athens, Greece: Morgan Kaufmann, Aug. 1997, pp. 426–435.

[9] B. Bustos, G. Navarro, and E. Chávez, "Pivot selection techniques for proximity searching in metric spaces," *Pattern Recognition Letters*, vol. 24, no. 14, pp. 2357–2366, Jan. 2003.

[10] P. Ciaccia, M. Patella, and P. Zezula, "A cost model for similarity queries in metric spaces," in *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'98)*. Seattle, WA: ACM Press, Jun. 1998, pp. 59–68.