A Theoretical Approach to the Self Similarity Join in a Distributed Enviroment

Claudio Gennaro
ISTI-CNR
via G. Moruzzi 1, 56124 Pisa, Italy
Email: claudio.gennaro@isti.cnr.it

***Abstract*—Efficient processing of similarity joins is important for a large class of data analysis and data-mining applications. This primitive finds all pairs of records within a predefined distance threshold of each other. However, most of the existing approaches have been based on spatial join techniques designed primarily for data in a vector space. Treating data collections as metric objects brings a great advantage in generality, because a single metric technique can be applied to many specific search problems quite different in nature.**

In this paper, we concentrate our attention on a special form of join, the *Self Similarity Join*, which retrieves pairs from the same dataset. In particular, we consider the case in which the dataset is split into subsets that are searched for self similarity join independently (e.g, in a distributed computing environment). To this end, we formalize the abstract concept of ϵ-*Cover*, prove its correctness, and demonstrate its effectiveness by applying it to two real implementations on a real-life large dataset.

I. Introduction

Similarity join between two datasets is a database primitive that retrieves all pairs of objects whose distance does not exceed a given threshold ϵ. This search paradigm has been successfully applied to a large class of applications such as data analysis, data mining, location-based applications, and time-series analysis. Moreover it has been generalized into a model in which a set of objects can only be pairwise compared through a distance measure satisfying the *metric space* properties [1].

The problem of similarity join emerges naturally in a variety of applications where the user is not only interested in the properties of single data objects but also in the properties of the data set as a whole, as, for instance, in data mining applications. Considering the typical clustering task of data mining, many of the state-of-the-art algorithms require to process all pairs of data items which have a distance not exceeding a user-given parameter ϵ. Consequently, many of these algorithms can be directly performed on top of a similarity join as proposed in [2].

An important special case, which is the focus of this research, is the self join where duplicate, near-duplicate, or very similar items within a single dataset are identified. Applications like data cleaning, data integration, and copy detection, typically rely on such a join. For instance, various Internet services often require an integration of heterogeneous sources of data. Such sources are typically unstructured whereas the intended services often require structured data. The main challenge is to provide consistent and error-free data, which implies the *data cleaning* [3], typically implemented by a sort of similarity join. The goal of data cleaning is to match records that refer to the same real-world entity while not being syntactically equivalent. In order to perform such a task, similarity rules are specified to decide whether specific pieces of data may actually be the same things or not. A similar approach can also be applied to the *copy detection* as, for instance, in applications that have been proposed to safeguard intellectual property [4].

Though the similarity join has always been considered as a basic similarity search operation, there are only few indexing techniques, most of them concentrating on vector spaces and designed for centralized systems. In this paper, we consider the problem from a much broader perspective and assume distance measures as metric functions. Such a view extends the range of possible data types to the multimedia dimension, which is typical for modern information retrieval systems.

However, the quadratic computational complexity of similarity joins prevents from applications on large data collections. To give an idea, let us consider a database of one million records. For computing the self similarity join, we need a number of distance evaluations of the order of one thousand billion. This scenario clearly requires an approach that exploits some form of preprocessing or data structure so that the query is answered efficiently. Recent studies [5], [6], [7] tried to approach the problem by recursively dividing the data until each partition contains few objects, at which point a nested-loop join (or some other optimized strategy) is used. In order to prevent that significant portion of the desired results will be missed since objects in one partition might need to be joined with objects in another partition, these techniques require that the data is replicated into the partitions.

Although this idea is quite intuitive, to the best knowledge of the author, its correctness has never been formally proven. Moreover, the data can be partitioned using a variety of techniques, such as ball partitioning, which divides the data on the basis of their distance to a single *pivot* (i.e., a random object), or the Voronoi-like partitioning which assigns every object to its closest pivot. For these reasons in this work we first formally define the abstract concept of ϵ-*Cover* that is valid in any continuous metric space and it is independent from the technique adopted for partitioning the dataset. We prove its correctness, and demonstrate its effectiveness by applying it to two real implementations.

The remainder of our paper is organized as follows: in Section II, we discuss related work. In Section III, we define principles of the similarity join search in metric spaces and give some definitions. In Section IV, we introduce the concept of ϵ-Cover and prove its suitability in solving the problem of similarity self join. Then, we introduce the problem of duplicates and propose an approach for dealing with it, in Section V. An experimental evaluation of our approach is presented in Section VI. Finally, in the conclusive section, we analyze the scalability of the proposed strategy. The proofs of the theorems are given in appendix A.

II. Related Work

The problem of similarity join has been studied in a variety of contexts. Most of the proposed approaches use specialized techniques for specific data types. For instance in [8] an approximate string join algorithm is proposed. This approach is based on segmenting strings into q-grams and introduces an additional overhead for building lists of q-grams. Another paper [9] proposes the *neighborhood join* algorithm, which uses the difference of lengths of compared strings as the simplified distance function for the pre-filtering rule.

Many techniques have been proposed on finding similar pairs of objects in high-dimensional spaces (e.g, [10], [11], [12], [13], [14]), while [15] offers a latest, comprehensive survey on spatial join techniques.

Concerning the problem of computing the join in parallel, other research studies propose methods of parallelizing high-dimensional proximity joins [16], [17], [18] and spatial joins [19], [20]. In [21] an approach to support similarity queries on structured data in a distributed system is presented. This system uses the edit distance for strings and the Euclidean distance for numerical values to express similarity and implements a form of similarity join

The extension of distributed data structures to support similarity search in metric spaces is studied in four different approaches recently proposed: the first two, the GHT* [22] and the VPT* [23], are native metric index structures whereas the other two, *MCAN* [24] and the M-Chord [25], transform the metric search issue into a different problem and take advantage of some well-known solutions.

As far as the author knows, this work represents the first attempt to formalize and study the problem of similarity join in metric space in a distributed environment. This work generalizes to any type of Cover the principle of overlapping partitions proposed in prior work [5] for self similarity joins in metric spaces using a Content-Addressable Network.

III. Preliminaries

In this section, we present the preliminaries for metric spaces and self similarity join algorithm, the notation we use in this work, and the notion of *self completeness up to* ϵ.

A. Metric spaces

Metric spaces formalize the notion of similarity by applying a metric function to decide the closeness of objects as a pair-wise distance, which can be seen as a measure of the objects *dis-similarity*. A *metric space* $\mathcal{M} = (\mathcal{D}, d)$ is defined by a domain of objects (elements, points) $\mathcal{D}$ and a total (distance) function d – a *non negative* ($d(x,y) \geq 0$ with $d(x,y) = 0$ iff $x = y$) and *symmetric* ($d(x,y) = d(y,x)$) function, which satisfies the *triangle inequality* ($d(x,y) \leq d(x,z) + d(z,y)$, $\forall x,y,z \in \mathcal{D}$).

Given a dataset of metric objects $X \subseteq \mathcal{D}$ and a query object $q \in \mathcal{D}$, two fundamental similarity queries can be defined. A *range query* retrieves all elements within distance r to q, that is the set $\{x \in X, d(q,x) \leq r\}$. A *k-nearest neighbor* query retrieves the k closest elements to q, that is a set $R \subseteq X$ such that $|R| = k$ and $\forall x \in R, y \in X - R, d(q,x) \leq d(q,y)$.

B. Partitions and Covers

Throughout the paper, we will often exploit the notions of *Partition* and *Cover* of a metric space $(\mathcal{D}, d)$, which are given in the following definitions

Definition 1: Given a set $X \subseteq \mathcal{D}$, a partition

$$\mathbf{P}(X) = \{\mathcal{P}_1(X), \ldots, \mathcal{P}_N(X)\}$$

is a collection of disjoint subsets whose union is X; that is, $\bigcup_{i=1}^{N} \mathcal{P}_i(X) = X$ and $\forall i \neq j\ \mathcal{P}_i(X) \cap \mathcal{P}_j(X) = \emptyset$.

The concept of cover is a generalization of the partition concept, in which we relax the constraint that subsets must be pairwise disjoint.

Definition 2: Given a set $X \subseteq \mathcal{D}$, a cover

$$\mathbf{C}(X) = \{\mathcal{C}_1(X), \ldots, \mathcal{C}_N(X)\}$$

is a collection of subsets whose union is X; that is, $\bigcup_{i=1}^{N} \mathcal{C}_i(X) = X$.
In practice, we are allowing the subsets $\mathcal{C}_i(X)$ to "overlap" one each other, $|\bigcup_{i=1}^{N} \mathcal{C}_i(X)| \leq \sum_{i=1}^{N} |\mathcal{C}_i(X)|$, in other words the objects in the original set X can be found replicated in more than one subset $\mathcal{C}_i(X)$.

In this paper, we will use a calligraphic capital symbol to denote a subset $\mathcal{P}_i(X) \in \mathbf{P}(X)$, and when we omit the argument X, it is just meant $X = \mathcal{D}$, the whole metric space. Partitions and covers will be typically defined through a boolean predicate that decides if an object $x \in \mathcal{D}$ belongs to a subset. Formally, the subset $\mathcal{P}_i(X) = \{x \in X \mid p_i(x) = \textit{true}\}$ where p_i is the boolean predicate defined as $p_i : \mathcal{D} \to \{\textit{true}, \textit{false}\}$. For instance, a closed ball $\mathcal{P}_b(X)$ can be defined as $\{x \in X \mid d(x, x_0) \leq r\}$, where r is a non negative real number and $x_0 \in \mathcal{D}$ the center of the ball. In this paper, subsets of $\mathcal{D}$ are referred to as *cells*.

Another important definition we need is the notion of cover inclusion.

Definition 3: Given two covers of the set X $\mathbf{C}(X) = \{\mathcal{C}_1(X), \ldots, \mathcal{C}_N(X)\}$ and $\mathbf{Q}(X) = \{\mathcal{Q}_1(X), \ldots, \mathcal{Q}_N(X)\}$, we say that cover $\mathbf{Q}$ includes $\mathbf{C}$, i.e., $\mathbf{C}(X) \subseteq \mathbf{Q}(X)$ if

$$\forall i\ \mathcal{C}_i(X) \subseteq \mathcal{Q}_i(X).$$

C. Self Similarity Join

Here we give the definition of self similarity join (SSJ):

Definition 4: The self similarity join up to ϵ (with $\epsilon > 0$) of a finite set of metric objects $X \subseteq \mathcal{D}$ is defined as the following collection of pairs

$$SSJ_\epsilon(X) \triangleq \{\langle x, y\rangle \in X \times X \mid d(x,y) \leq \epsilon\}$$

We now come to the central definition of our approach.

Definition 5: We say that the cover $\mathbf{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_N\}$ *is self complete up to* ϵ if

$$\forall X \subseteq \mathcal{D}: \qquad SSJ_\epsilon(X) = \bigcup_{i=1}^{N} SSJ_\epsilon(\mathcal{C}_i(X))$$

In practice, this property ensures that if we compute the SSJ up to ϵ on each subset $\mathcal{C}_i(X)$, it is guaranteed that we do not miss any pair that is in the SSJ applied to the whole dataset X. As it is easy to understand, this is not true in general, since objects in one subset might need to be joined with objects in another subset. It is trivial to prove that any partition $\mathbf{P}$ is guaranteed to be self complete up to 0.

IV. $\epsilon-$COVER

In this section we define a special type of cover: the $\epsilon-$Cover of a continuous metric space $(\mathcal{D}, d)$. In general $\mathcal{D}$ can be any subspace of continuous metric space $\mathcal{S}$, in this case we only require that $\mathcal{D}$ is convex[1].

Definition 6: Given an arbitrary partition $\mathbf{P}= \{\mathcal{P}_1, \ldots, \mathcal{P}_N\}$ of the metric space $(\mathcal{D}, d)$, the corresponding $\epsilon-$Cover $O^\epsilon(\mathbf{P}) = \{\mathcal{P}^\epsilon_1, \ldots, \mathcal{P}^\epsilon_N\}$ is defined as

$$\mathcal{P}^\epsilon_i \triangleq \{x \in \mathcal{D} \mid d(x, \mathcal{P}_i) \leq \frac{\epsilon}{2}\} \tag{1}$$

Where the distance between a set and an object is defined as $d(x, X) = \inf\{d(x,y) \mid y \in X\}$, as the intuition suggests.
In practice, this technique consists of defining a basic partition $\{\mathcal{P}_1, \ldots, \mathcal{P}_N\}$ and a corresponding cover in which we expand the sets of a "strip" of width $\epsilon/2$ from which the i-th subset borrows objects from the k-th subset. Figure 1 shows an illustration of a $\epsilon-$Cover in $\mathbb{R}^2$. It is worth noting that, in principle a subset $\mathcal{P}_i$ is permitted to be open, i.e., can contain only part of its boundary[2]. Therefore in general $O^0(\mathbf{P}) \neq \mathbf{P}$ is true.

Theorem 1: Any cover $\mathbf{Q}$ including an ϵ-Cover, i.e. $O^\epsilon(\mathbf{P}) \subseteq \mathbf{Q}$, is self complete up to ϵ.

Now that we have given the notion of $\epsilon-$Cover and proved its suitability for the problem of SSJ, we are going, in the next sections, to show two real examples of covers. In order to prove that these approaches are self complete up to ϵ, it is sufficient to prove that they include the general $\epsilon-$Cover of Definition 6.

[1]In general, a metric space is said to be convex if for any $x \neq y$ there exists a point z different from x and y with $d(x,y) = d(x,z)+d(z,y)$ [26].

[2]Suppose $\mathcal{D}$ is a metric space, X is a subset of $\mathcal{D}$ and $x \in \mathcal{D}$. Then x is called a boundary point of X in $\mathcal{D}$ if, and only if, $d(x,X) = 0 = d(a, \overline{X})$. Where $\overline{X}$ is the complement of X in $\mathcal{D}$ [27].

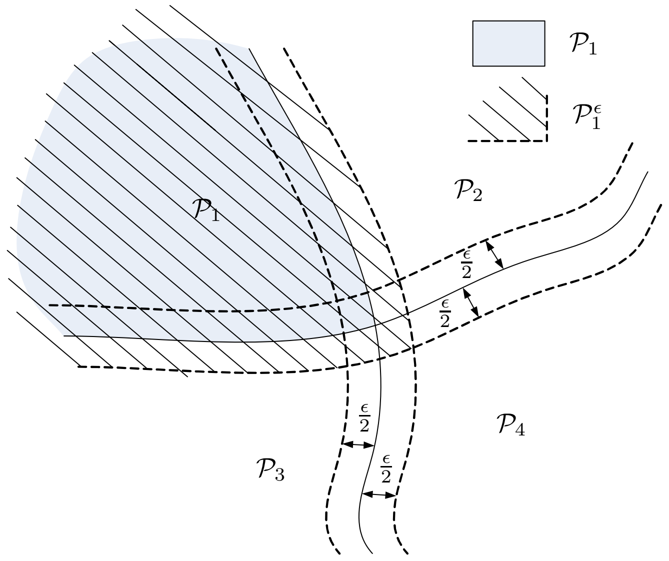

Fig. 1. Example of $\epsilon-$Cover in $\mathbb{R}^2$ including four cells.

A. $\epsilon-$Voronoi Cover

The $\epsilon-$*Voronoi Cover* is based on a generalization of the well–known homonymous tessellation (sometimes known as Dirichlet tessellation). This approach is similar to the one used in the Quickjoin technique presented by Jacox et al. in [6].

Definition 7: Given a set of N distinct reference objects $v_1, \ldots, v_N$ of $\mathcal{D}$. The Voronoi partition $\mathbf{V}_N = \{\mathcal{V}_1, \ldots, \mathcal{V}_N\}$ is defined as in the following:

$$\mathcal{V}_i \triangleq \dot{\mathcal{V}}_i \bigcup \widehat{\mathcal{V}}_i. \tag{2}$$

Where

$$\dot{\mathcal{V}}_i \triangleq \{x \in \mathcal{D} \mid d(x, v_i) < d(x, v_j) \ \forall j \neq i\}, \tag{3}$$

and

$$\widehat{\mathcal{V}}_i \triangleq \{x \in \mathcal{D} \mid d(x, v_i) = d(x, v_j) \ \forall j \geq i\}. \tag{4}$$

Note that $\dot{\mathcal{V}}_i$ represents the interior of a cell of the Voronoi decomposition [28] and $\widehat{\mathcal{V}}_i$ is its boundary, in fact the condition in Eq. (4) guarantees that any object $x \in \mathcal{D}$ belongs to just one cell $\mathcal{V}_i$.

Definition 8: Given a Voronoi partition $\mathbf{V}_N = \{\mathcal{V}_1, \ldots, \mathcal{V}_N\}$ the corresponding $\epsilon-$Voronoi Cover $\mathbf{V}^\epsilon_N$ is a collection of cells $\{\mathcal{V}^\epsilon_1, \ldots, \mathcal{V}^\epsilon_N\}$ for which

$$\mathcal{V}^\epsilon_i \triangleq \begin{cases} \mathcal{V}_i & \text{if } \epsilon = 0 \\ \mathcal{V}_i + \bigcup_{\forall k \neq i} \mathcal{V}^\epsilon_{ik} & \text{if } \epsilon > 0 \end{cases} \tag{5}$$

where the cells $\mathcal{V}^\epsilon_{ik}$ are defined as

$$\mathcal{V}^\epsilon_{ik} \triangleq \{x \in \mathcal{V}_k \mid d(x, v_i) - d(x, v_k) \leq \epsilon\} \quad i \neq k, \tag{6}$$

Figure 2 illustrates an example of $\epsilon-$Voronoi Cover on $\mathbb{R}^2$ including three cells. Note that the cells $\mathcal{V}_i$ represent the typical Voronoi diagram, while, $\mathcal{V}^\epsilon_i$ can be seen as an extension of a single cell for a "strip" of width $\geq \epsilon/2$.

Theorem 2: Any $\epsilon-$Voronoi Cover $\mathbf{V}^\epsilon_N$ is self complete up to ϵ.

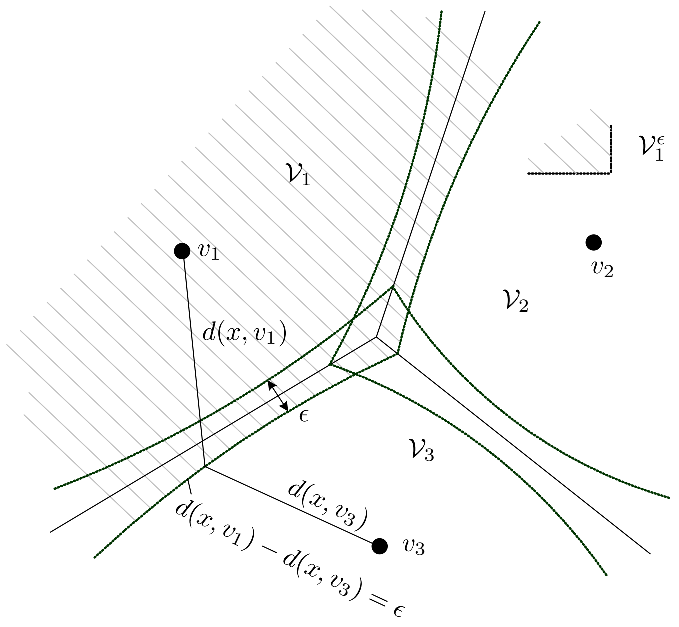

Fig. 2. Example of $\epsilon-$Voronoi Cover in $\mathbb{R}^2$ including three cells.

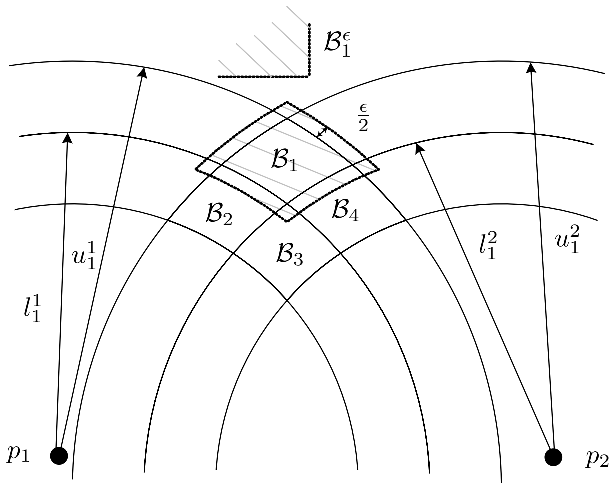

Fig. 3. Example of $\epsilon-$Box Cover in $\mathbb{R}^2$ including four cells.

B. $\epsilon-$Box Cover

$\epsilon-$Box Cover is based on a cell obtained by intersecting m ring-shaped regions, which assumes the form of hyper-rectangle in the well-known transformed "pivot space".

Definition 9: Given a set of m distinct reference objects $b_1,\ldots,b_m$ of $\mathcal{D}$, the Box partition $\mathbf{B}_N$ is a collection of cells $\{\mathcal{B}_1,\ldots,\mathcal{B}_N\}$, where the cell $\mathcal{B}_i$ (i.e., box), is given by the cartesian product of closed interval of distances with b_j. More precisely, each box $\mathcal{B}_i$ is represented by two sets of m non negative real numbers $L^i = \{l_1^i,\ldots,l_m^i\}$ and $U^i = \{u_1^i,\ldots,u_m^i\}$ be such that $l_j^i < u_j^i$, and is defined as

$$\mathcal{B}_i = \{x\in\mathcal{D},\ 1\le j\le m \mid l_j^i \le d(x,b_j) < u_j^i\}. \tag{7}$$

The sets L_i and U_i are chosen in order to guarantee that[3]

$$\bigcup_{i=1}^{N}\mathcal{B}_i = \mathcal{D} \qquad \mathcal{B}_i\bigcap\mathcal{B}_k=\emptyset\ \forall i\ne k \tag{8}$$

Definition 10: Given a Box partition $\mathbf{B}_N=\{\mathcal{B}_1,\ldots,\ \mathcal{B}_N\}$ the corresponding $\epsilon-$Box Cover $\mathbf{B}_N^\epsilon$ is a collection of cells $\{\mathcal{B}_1^\epsilon,\ldots,\ \mathcal{B}_N^\epsilon\}$ for which

$$\mathcal{B}_i^\epsilon \triangleq \begin{cases} \mathcal{B}_i & \text{if } \epsilon=0 \\ \mathcal{B}_i+\bigcup_{\forall k\ne i}\mathcal{B}_{ik}^\epsilon & \text{if } \epsilon>0 \end{cases} \tag{9}$$

where the cell $\mathcal{B}_{ik}^\epsilon$ is defined as follows ($i\ne k<N$)

$$\mathcal{B}_{ik}^\epsilon = \{x\in\mathcal{B}_k,\ 1\le j\le m \mid l_j^i-\frac{\epsilon}{2}\le d(x,b_j)\le u_j^i+\frac{\epsilon}{2}\}. \tag{10}$$

Figure 3 illustrates the idea behind of the above definitions.

Theorem 3: Any $\epsilon-$Box Cover $\mathbf{B}_N^\epsilon$ is self complete up to ϵ.

[3]Note that, u_j^N is permitted to be ∞.

V. Duplicate Avoidance

An important issue arises in the application of the ϵ-Cover: the problem of duplicate pairs in the result of the join query. The problem of duplicates detection is, in general, not a complex task, since the process that collects the result set can easily identify duplicates provided that each object is uniquely identified by a number. However, in case the pairs retrieved must be sent over the network to a collector processing node, this overhead can affect the performance of the whole SSJ response time.

In this section, we try to approach this problem by building a special boolean predicate $r_i(x,y)$ for each cell $\mathcal{C}_i$ that permits to decide if a qualified $\langle x,y\rangle$ must be reported or not, guaranteeing in this way that $\langle x,y\rangle$ is returned only once. It is worth nothing that this technique is valid for any cover no matter is ϵ-Cover or not (it does not guarantee correctness but just absence of duplicates). If you are not interested in the details of how these functions are implemented, skip now to the next section.

Before defining $r_i(x,y)$, we need the following two definitions.

Definition 11: Given a cover $\mathbf{C}=\{\mathcal{C}_1,\ldots,\mathcal{C}_N\}$, we define the bijective function

$$\alpha(C_i):\mathbf{C}\rightarrow\{1,\ldots,N\}$$

that assigns a distinct integer in the interval [1,N] to each cell of the cover.

Definition 12: for each object $x\in\mathcal{D}$ we define the function $\sigma(x)$ as follows:

$$\sigma(x):\mathcal{D}\rightarrow\Theta(\{1,2,\ldots,N\})$$

where $\Theta(\{1,2,\ldots,N\})$ is the collection of all possible subsets of $\{1,2,\ldots,N\}$ excluding the empty set, and $\sigma(x)$ has the following property

$$x\in\mathcal{C}_i \Leftrightarrow \alpha(C_i)\in\sigma(x) \tag{11}$$

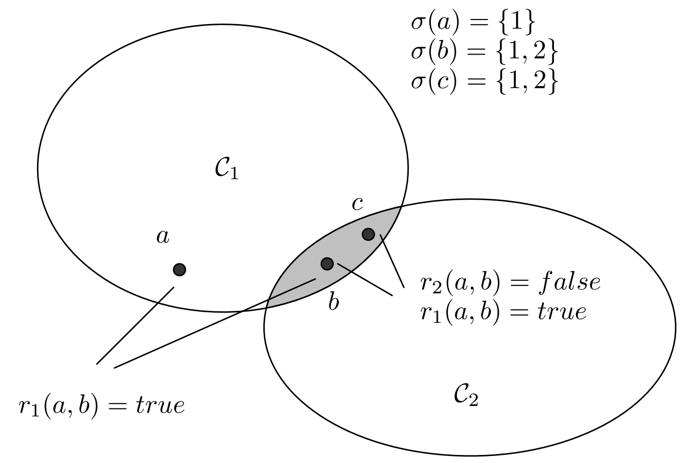

Fig. 4. Example of $\epsilon-$Cover in $\mathbb{R}^2$ including four cells.

In practice, $\sigma(x)$ is a function that returns the set of indexes of the covers to which x belongs.

Definition 13: Given a cover $\mathbf{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_N\}$, for each cell $\mathcal{C}_i$ and a pair of object $x, y \in \mathcal{C}_i$, the function $r_i(x, y) : \mathcal{D} \times \mathcal{D} \to \mathbb{B}$ is defined as:

$$r_i(x,y) = \begin{cases} \textit{true} \text{ if } \min \sigma(x) \bigcap \sigma(y) = i \\ \\ \textit{false} \text{ otherwise} \end{cases} \quad (12)$$

We can finally avoid duplicates in elaborating the self similarity join by using the procedure SSJ^*_ϵ given in the following definition.

Definition 14: Let $X \in \mathcal{D}$ be a finite dataset of metric objects and $\mathbf{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_N\}$ be an arbitrary cover. Assuming $X_i = \mathcal{C}_i(X)$, the $SSJ^*_\epsilon(X_i)$ is defined as in the following

$$\begin{aligned} SSJ^*_\epsilon(X_i) \triangleq \{&\langle x, y\rangle \in X_i \times X_i \mid \\ & d(x,y) \leq \epsilon \ \wedge \ r_i(x,y) = \textit{true}\}. \end{aligned} \quad (13)$$

The idea behind the formal definitions is straightforward. When a qualified pair $\langle x, y\rangle$ is found in subset X_i the corresponding function $r_i(x, y)$ returns *true* either if the pair $\langle x, y\rangle$ is contained only in X_i or i is the smallest index of the subsets that share the same pair $\langle x, y\rangle$. See Figure 4. This concept is formalized by the following theorem.

Theorem 4: Let $X \in \mathcal{D}$ be an arbitrary finite dataset of metric objects and let $\mathbf{C}(X) = \{\mathcal{C}_1(X), \ldots, \mathcal{C}_N(X)\}$ be a generic cover, then

$$\forall i \neq j \ SSJ^*_\epsilon(X_i) \bigcap SSJ^*_\epsilon(X_j) = \emptyset,$$

where $X_i = \mathcal{C}_i(X)$ and $X_j = \mathcal{C}_j(X)$.

VI. Performance Evaluation

A. Datasets and Problem Size

In order to test the Voronoi and Box methodologies presented above, we have conducted some experiments using a large real-life dataset of MPEG-7 Scalable Color Descriptors

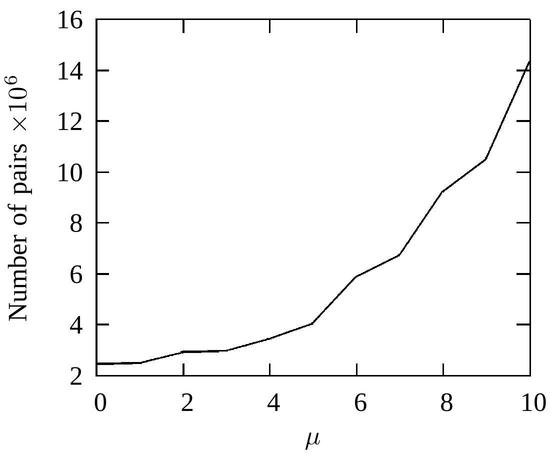

Fig. 5. Performance evaluation of a sequential implementation of NL algorithm.

(SCD) extracted from one million images of the CoPhIR dataset [29]. The distance used for this visual descriptor is the L_1, as suggested by the MPEG-7 standard [30].

In the paper, we measure the computational complexity through the number of distance evaluations. Assuming M be the dataset size, the self similarity join can be evaluated by a simple algorithm that computes $\frac{M \cdot (M-1)}{2}$ distances between all pairs of objects, referred in literature to as *Nested Loop* (NL).

The challenge of the proposed approach is to address the problem of the intrinsic quadratic complexity of similarity joins, with the aim of limiting the computation time, by involving an increasing number of computational nodes as the dataset size grows. Indeed, consider that the time estimated for the simple NL algorithm in sequential manner is more than five days for the complete dataset of one million objects. Figures 5 shows the number of pairs retrieved for increasing values of ϵ for the dataset of one million objects. In all the proposed experiments we implicitly assume to assign each cell to an independent computational node.

The main aim of this experimentation is to study the suitability of these methodologies to cope with the problem of SSJ, with special regard to the scalability issues.

It is important to remark that, in a real scenario as the one we are evaluating, the calculation of the distance function d has typically a high computational cost. Therefore, the main objective of a metric-based data structure is to reduce the number of distance computations at query time. The number of distance computations is typically considered as an indicator of the structure efficiency. On the other hand, during the evaluation of the SSJ^*_ϵ on a subset $X_i = \mathcal{P}_i(X)$ by means of the NL algorithm, we can employ the knowledge of the precalculated distances with respect to the reference objects to get a lower bound of the distance $d(x, y)$. In case this lower bound exceeds the threshold ϵ we can discard the pair $\langle x, y\rangle$ without computing $d(x, y)$. This technique is often known as *pivot filtering*. For this reason, to give a fair comparison, in the experimental evaluation we also report the number of pairs of the subset X_i processed by NL, which is given by $\approx n^2/2$, where $n = |X_i|$.

The latter amount can be seen as an upper bound for the computational cost of the SSJ, while the former one (pivot filtering) represents in some way a lower bound for the computational cost. The actual computational time will fall in between these two figures, in fact this cost may also depend on the cost for pivot filtering, the cost for implementing the duplicate avoidance. Moreover, in many cases metric access methods can be employed, allowing us to obtain an average run time complexity near to $O(n \log n)$, when ϵ is small.

Concerning the global costs, we use the following two characteristics to measure the computational costs of a query:

- *sequential distance computations* – the sum of the number of distance computations of SSJ_ϵ^* on all subsets X_i,
- *parallel distance computations* – the maximal number of distance computations among the SSJ_ϵ^* performed on the subsets X_i in parallel.

To give an example, consider a case of just three subsets on which we execute a SSJ_ϵ^* with number of distance computations being respectively 200, 300, and 500. In this case, the total distance computations would be 1,000 and the parallel distance computations would be 500.

In this experimental evaluation we have build a ϵ-Voronoi cover of cardinality $N = 200$ (i.e., employing 200 reference objects), and a ϵ-Box cover of cardinality $N = 128$ that employs 10 reference objects. In order to choose the reference objects, we use the Incremental Selection algorithm described in [31].

In Figure 6, we report the sequential costs of the SSJ using the two approaches, in which we set ϵ to 10 and we varied the SSJ threshold from 0 up to ϵ. In fact, the number of distance computations can be significantly lower when the SSJ threshold does not reach maximum value ϵ supported by the covers.

As it can be seen, in the two cases, the performance of Box cover is much better than the one obtained using Voronoi cover. The rationale for this behavior mainly lies in the fact the Box cover uses all 10 reference objects when employing pivot filtering, while in Voronoi cover we can only exploit one reference object in each cell.

This discrepancy is even more pronounced if we consider the experiments concerning the parallel costs of Figure 7. This is due to the fact that in considering the parallel costs, the uniformness of the objects distribution over the cells strongly affects the performance.

In this second set of experiments, we analyze the behavior of the two methodologies by concentrating our attention on scalability issues. Particularly, we have considered four collections of objects (i.e., SCD) starting from 125 thousands of objects and growing by doubling its size, that is 125,000, 250,000, 500,000, and 1000,000. The number of (computational) cells is doubled accordingly and it is 25, 50, 100, and 200 for the Voronoi covers and 16, 32, 64, and 128 for the Box cover. Figure 8 shows the corresponding graphs for a SSJ

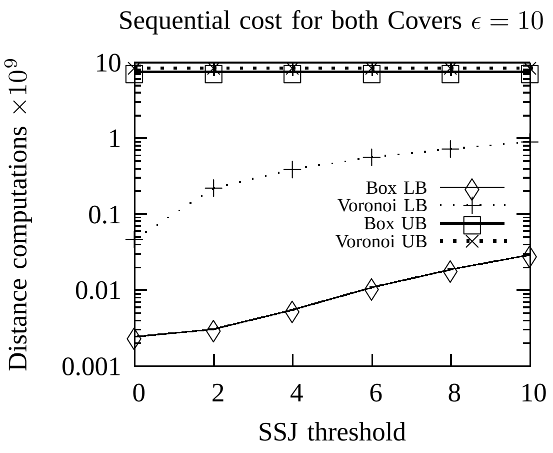

Fig. 6. Sequential cost of *SSJ* for several values of the threshold and for the two types of covers with $\epsilon = 10$ (LB = “lower bound”, UB = “upper bound”).

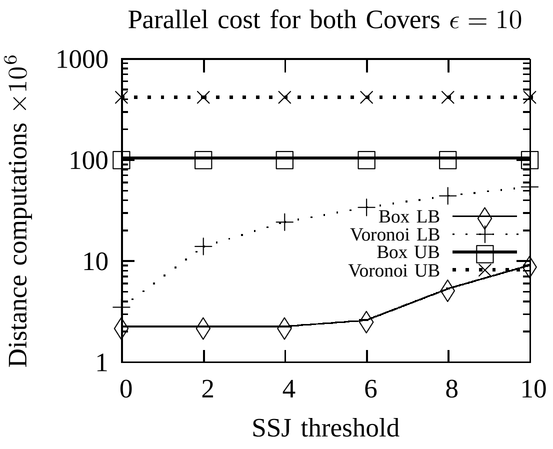

Fig. 7. Parallel cost of *SSJ* for several values of the threshold and for the two types of covers with $\epsilon = 10$ (LB = “lower bound”, UB = “upper bound”).

with threshold equal to 10. We can see that in both cases the costs grow linearly (both axis have logarithmic scales). We will come back to this point in the conclusive section.

We also studied the replication factor R of the covers, which is defined as the ratio M^*/M, where M is the size of the dataset and M^* the sum of cardinalities of all subsets, i.e., $\sum |X_i|$. Therefore, $R \geq 1$ ($R = 1$ means no duplicates).

Figure 9 shows the replication factor scalability, for the same four datasets of the previous experiments. As it is possible to note, the worst case is exhibited by the Voronoi cover. However, it must be highlighted that the extra space due to replication is scattered over an increasing number of cells. Therefore, in a distribute environment this drawback is tolerable.

Another interesting issue we can study is the degree of overlapping among the subsets, which we call *overlapping factor*. It is defined as the number of subsets that share

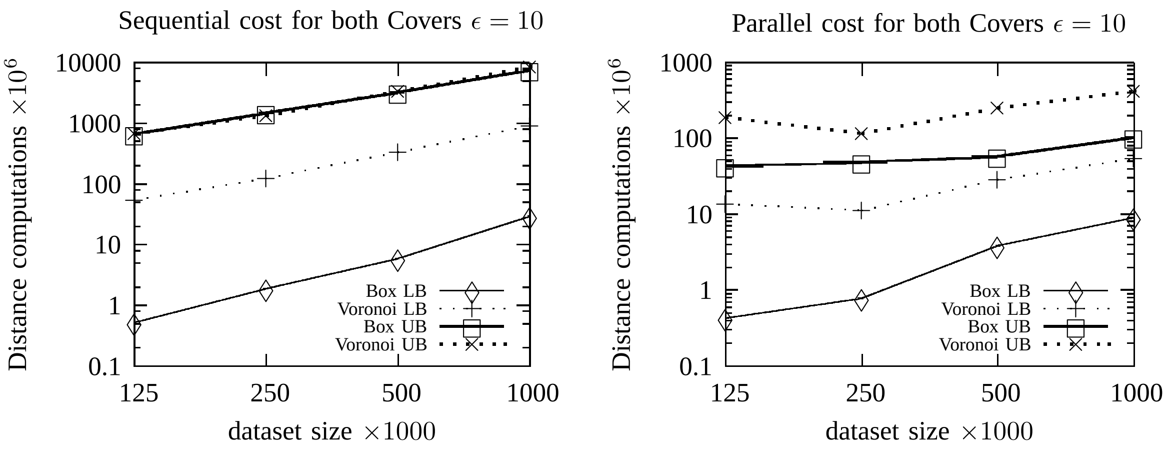

Fig. 8. Sequential (left) and parallel (right) cost for the two type of covers with $\epsilon = 10$, for a *SSJ* with threshold 10 (LB = "lower bound", UB = "upper bound").

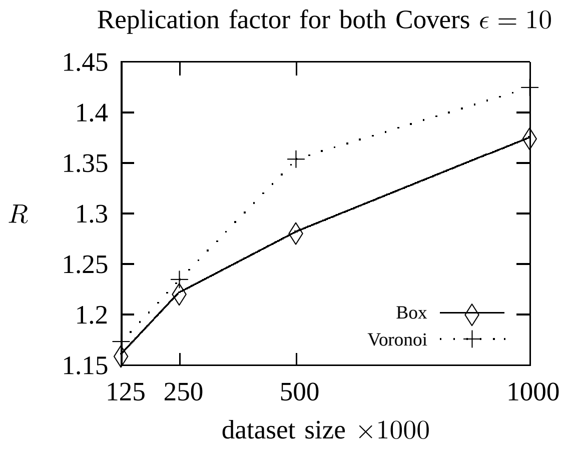

Fig. 9. Replication factor for the two type of covers with $\epsilon = 10$.

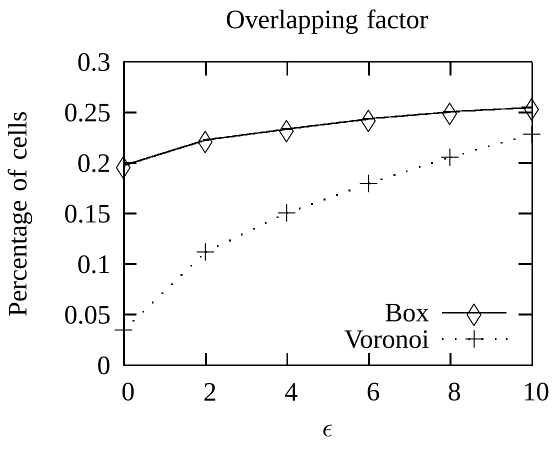

Fig. 10. Overlapping factor the two type of covers vs ϵ.

duplicate objects with a given subset, divided by the total number of subsets. Figure 10 shows the average overlapping factor over all subsets of a cover. This metric allows us to see the percentage of the nodes (cells) that will be involved during the creation of the network in a possible distributed implementation of the ϵ-Cover. If, for instance, we have that the overlapping factor is the 20% for a given ϵ-Cover this means that during the insertion of an objects a cell must communicate with the 20% of the network in order to allow the overlapping cells to replicate it.

VII. Scalability Analysis and Conclusion

The replication factor and especially the non-uniform distribution of the objects among the cells can strongly affect the scalability of these types of strategies that share the idea of exploiting partition redundancy [5], [6], [7] to perform similarity join. In principle, if we uniformly distribute a dataset of size M into N cells, the time for performing the SSJ employing the NL strategy should be $O(N\left(\frac{M}{N}\right)^2) = O(\frac{M^2}{N})$ for the sequential approach and $O(\frac{M^2}{N^2})$ for the parallel approach. Therefore if dataset size scales as k and we correspondingly expand the cover cardinality by the same factor k, the total sequential time should scale linearly with k and the parallel time should remain constant, which is not exactly the case of Figure 8.

This issue can be studied by the experiment of Figure 11, in which we estimate the theoretical limits of the size scalability of any type ϵ-Cover strategy of our dataset for different values of ϵ.

The estimate is obtained with the following heuristic strategy. For each object of the dataset we evaluate the number of neighbors objects within a range of ϵ. We refer to these clusters as *atomic-clusters*, since they will be reside entirely (due to overlaps) in a cell of a any ϵ-Cover. The size of the greatest atomic-cluster, $n_{max}(\epsilon)$, determines the limit of the parallel time. Assuming for instance 20,000 be the maximum size we

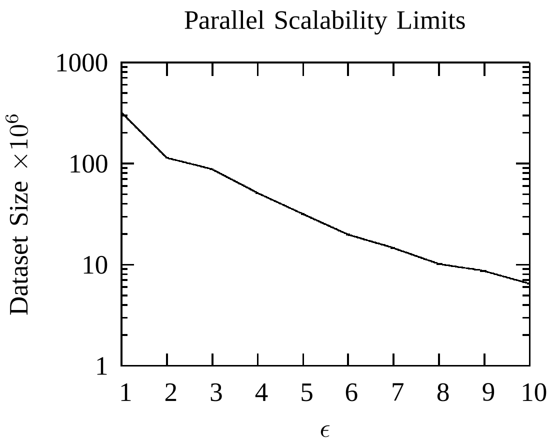

Fig. 11. Scalability Limits of covers vs ϵ.

tolerate for a cell (which experimentally corresponds to about 2 minutes for the NL strategy) and hypothesizing a linear scale-up of the size of the atomic-clusters with increasing number of objects in the dataset, we obtain the scalability limit of $20,000/n_{max}(\epsilon) * 1,000,000$ shown in Figure 11. This experiment reveals that for instance for $\epsilon = 10$, (for this type of dataset, i.e., SCD) it is hard to scale (in parallel) beyond 6 million objects.

Although the results presented in this paper shown the intrinsic constraints of the ϵ-Cover strategies, we believe that similarity join in metric spaces is the necessary complement of more famous similarity range and nearest neighbor search primitives. In view of the fact that, a scenario where a specialized infrastructure created just to support only similarity join is far to be realistic. Moreover, the replication will have a positive impact to the number of cells involved during the range and nearest neighbor query processing.

ACKNOWLEDGMENT

This work has been partially supported by the SAPIR (Search on Audiovisual content using P2P IR) project, funded by the European Commission under IST FP6 (Sixth Framework Programme, Contract no. 45128) and by the NeP4B (Networked Peers for Business) FIRB project, funded by the Italian Council.

REFERENCES

[1] E. Chávez, G. Navarro, R. Baeza-Yates, and J. L. Marroquín, “Proximity searching in metric spaces,” 1999, to appear in *ACM Computing Surveys*.

[2] C. Böhm, B. Braunmüller, M. Breunig, and H.-P. Kriegel, “High performance clustering based on the similarity join,” in *CIKM ’00*, 2000, pp. 298–305.

[3] M.-L. Lee, T. W. Ling, H. Lu, and Y. T. Ko, “Cleansing data for mining and warehousing,” in *Database and Expert Systems Applications*, 1999, pp. 751–760. [Online]. Available: citeseer.ist.psu.edu/lee99cleansing.html

[4] S. Brin, J. Davis, and H. García-Molina, “Copy detection mechanisms for digital documents,” 1995, pp. 398–409. [Online]. Available: citeseer.ist.psu.edu/brin95copy.html

[5] C. Gennaro, “A Content-Addressable Network for Similarity Join in Metric Spaces,” in *Proceedings of the Third International Conference on Scalable Information Systems (Infoscale 2008)*. ACM Press, June 2008.

[6] E. H. Jacox and H. Samet, “Metric space similarity joins,” *ACM Trans. Database Syst.*, vol. 33, no. 2, 2008.

[7] V. Dohnal, C. Gennaro, and P. Zezula, “Similarity Join in Metric Spaces Using eD-Index,” in *Proc. of the14th International DEXA Conference*, ser. LNCS, vol. 2736. Springer, May 2003, pp. 484–493.

[8] H. Galhardas, D. Florescu, D. Shasha, E. Simon, and C.-A. Saita, “Declarative data cleaning: Language, model, and algorithms,” in *VLDB*, 2001, pp. 371–380.

[9] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava, “Approximate string joins in a database (almost) for free,” in *VLDB 2001*, 2001, pp. 491–500. [Online]. Available: citeseer.ist.psu.edu/article/gravano01approximate.html

[10] D. V. Kalashnikov and S. Prabhakar, “Fast similarity join for multi-dimensional data,” vol. 32, no. 1, pp. 160–177, 2007.

[11] C. Li, L. Jin, and S. Mehrotra, “Supporting efficient record linkage for large data sets using mapping techniques,” *World Wide Web*, vol. 9, no. 4, pp. 557–584, 2006.

[12] T. Brinkhoff, H.-P. Kriegel, and B. Seeger, “Efficient processing of spatial joins using r-trees,” in *SIGMOD ’93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 1993, pp. 237–246.

[13] A. Corral, Y. Manolopoulos, Y. Theodoridis, and M. Vassilakopoulos, “Closest pair queries in spatial databases,” *SIGMOD Rec.*, vol. 29, no. 2, pp. 189–200, 2000.

[14] N. Koudas and K. C. Sevcik, “Size separation spatial join,” in *Proc. of the ACM SIGMOD Conference on Management of Data*, 1997, pp. 324–335.

[15] E. H. Jacox and H. Samet, “Spatial join techniques,” *ACM Trans. Database Syst.*, vol. 32, no. 1, p. 7, 2007.

[16] J. C. Shafer and R. Agrawal, “Parallel algorithms for high-dimensional similarity joins for data mining applications,” in *VLDB*. Morgan Kaufmann, 1997, pp. 176–185.

[17] K. Alsabti, S. Ranka, and V. Singh, “An efficient parallel algorithm for high dimensional similarity join,” *ipps*, vol. 00, p. 0556, 1998.

[18] N. Koudas and K. C. Sevcik, “High dimensional similarity joins: Algorithms and performance evaluation,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 1, pp. 3–18, 2000.

[19] X. Zhou, D. J. Abel, and D. Truffet, “Data partitioning for parallel spatial join processing,” in *SSD*, 1997, pp. 178–196.

[20] E. G. Hoel and H. Samet, “Data-parallel spatial join algorithms,” in *ICPP ’94: Proceedings of the 1994 International Conference on Parallel Processing*. Washington, DC, USA: IEEE Computer Society, 1994, pp. 227–234.

[21] M. Karnstedt, K.-U. Sattler, M. Hauswirth, and R. Schmidt, “Cost-aware processing of similarity queries in structured overlays,” *p2p*, vol. 0, pp. 81–89, 2006.

[22] M. Batko, C. Gennaro, and P. Zezula, “A Scalable Nearest Neighbor Search in P2P Systems,” in *Proc. of the the 2nd International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P 2004)*, ser. Lecture Notes in Computer Science, vol. 3367. Springer, 2005, pp. 79–92.

[23] M. Batko, F. Falchi, D. Novák, and P. Zezula, “On scalability of the similarity search in the world of peers,” in *INFOSCALE ’06: Proceedings of the First International Conference on Scalable Information Systems*. IEEE Computer Society, 2006.

[24] F. Falchi, C. Gennaro, and P. Zezula, “A Content Addressable Network for Similarity Search in Metric Spaces,” in *Proc. of the 2nd DBISP2P Workshop*, ser. LNCS, vol. 4125. Springer, 2005, pp. 98–110.

[25] D. Novák and P. Zezula, “M-chord: A scalable distributed similarity search structure,” in *INFOSCALE ’06: Proceedings of the First International Conference on Scalable Information Systems*. IEEE Computer Society, May 2006.

[26] I. Kaplansky, *Set Theory and Metric Spaces*. AMS Bookstore, 2001.

[27] M. Ó. Searcóid, *Metric Spaces*. Springer, 2006.

[28] H. Edelsbrunner, *Algorithms in combinatorial geometry*. New York, NY, USA: Springer-Verlag New York, Inc., 1987.

[29] P. Bolettieri, A. Esuli, F. Falchi, C. Lucchese, R. Perego, and F. Rabitti, “Enabling content-based image retrieval in very large digital libraries,” in *Second Workshop on Very Large Digital Libraries (VLDL 2009), 2 October 2009, Corfu, Greece*. Pisa, Italy: DELOS, 2009, pp. 43–50.

[30] "Mpeg requirements group, mpeg-7 overview," 2003, doc. ISO/IEC JTC1/SC29/WG11N5525.

[31] B. Bustos, G. Navarro, and E. Chvez, "Pivot selection techniques for proximity searching in metric spaces," in *Proc. of the XXI Conference of the Chilean Computer Science Society (SCCC'01)*, 2001, pp. 33–40.

APPENDIX A
THEOREMS PROOFS

A. Proof of the Theorem 1

Proof: Let us suppose that we have an arbitrary pair of objects $x, y \in \mathcal{D}$ such that $d(x, y) \leq \epsilon$, we must prove that there exists at least a subset $\mathcal{Q}_i^\epsilon$ for which $x, y \in \mathcal{Q}_i^\epsilon$. However since $\mathcal{Q}_i^\epsilon \supseteq \mathcal{P}_i^\epsilon$ it is sufficient to prove that there exists a $\mathcal{P}_i^\epsilon$ such that $x, y \in \mathcal{P}_i^\epsilon$. Since for definition of partition the union of all sets $\mathcal{P}_i$ is $\mathcal{D}$, the objects x and y can either fall in distinct sets or in a single set. Let us consider the former case, i.e., in which $x \in \mathcal{P}_i$ and $y \in \mathcal{P}_j$ with $i \neq j$ (if they were not distinct then the pair is already included in a single partition). Moreover, since for definition $\mathcal{D}$ is convex and continuous, there exists an object z such that $d(x, z) = d(y, z) = d(x, y)/2$. Since $d(x, y) \leq \epsilon$, we have that $d(x, z) \leq \epsilon/2$ and $d(y, z) \leq \epsilon/2$. Consequently, there can be three cases, $z \in \mathcal{P}_i$, $z \in \mathcal{P}_j$, or $z \in \mathcal{P}_k$ (with $k \neq i$, $k \neq j$). From definition of distance between a set and an object we have that: in the first (second) case since $z \in \mathcal{P}_i$ ($z \in \mathcal{P}_j$) and $d(y, z) \leq \epsilon/2$ ($d(x, z) \leq \epsilon/2$), this implies that $y \in \mathcal{P}_i^\epsilon$ ($x \in \mathcal{P}_j^\epsilon$) and hence both x, y lie on $\mathcal{P}_i^\epsilon$ ($\mathcal{P}_j^\epsilon$); in the third case, $x \in \mathcal{P}_k^\epsilon$ and $y \in \mathcal{P}_k^\epsilon$, hence, both x and y lie on $\mathcal{P}_k^\epsilon$. ■

B. Proof of the Theorem 2

Proof: In order to prove this theorem, we must verify that $\mathbf{V}_N^\epsilon$ includes the general $\epsilon-$Cover of $\mathbf{V}_N$, i.e., that $\mathbf{V}_N^\epsilon \supseteq O^\epsilon(\mathbf{V}_N)$ is always true, which can be translated in the following condition:

$$x \in \mathcal{P}_i^\epsilon \Rightarrow x \in \mathcal{V}_i^\epsilon, \tag{14}$$

where $\mathcal{P}_i^\epsilon \in O^\epsilon(\mathbf{V}_N)$ is defined as in (1), i.e.,

$$\mathcal{P}_i^\epsilon \triangleq \{x \in \mathcal{D} \mid d(x, \mathcal{V}_i) \leq \frac{\epsilon}{2}\} \tag{15}$$

From (5) condition (14) can be expressed as

$$d(x, \mathcal{V}_i) \leq \frac{\epsilon}{2} \Rightarrow \exists k \neq i \mid x \in \mathcal{V}_{ik}^\epsilon \vee x \in \mathcal{V}_i,$$

which can be also written as:

$$x \notin \mathcal{V}_i \wedge x \notin \mathcal{V}_{ik}^\epsilon \ \forall k \neq i \Rightarrow d(x, \mathcal{V}_i) > \frac{\epsilon}{2}.$$

When $x \notin \mathcal{V}_i$ and $x \notin \mathcal{V}_{ik}^\epsilon$ from (6) we have that

$$d(x, v_i) - d(x, v_k) > \epsilon. \tag{16}$$

However, since the space is continuous there must exist an object z for which

$$d(z, x) = \inf\{d(x, w) \mid d(w, v_i) = d(w, v_k)\}.$$

The object z represents the nearest "point" of the generalized hyperplane that separates cells $\mathcal{V}_i$ and $\mathcal{V}_k$ to an arbitrary object x. Therefore, the distance $d(x, z)$ is the distance between the boundary of $\mathcal{V}_i$ and x. Using the triangle inequality we know that

$$d(z, v_i) + d(x, z) \geq d(x, v_i), \tag{17}$$

and that

$$d(z, v_k) - d(x, z) \leq d(x, v_k). \tag{18}$$

By combining inequalities (17) and (18) with inequality (16), we obtain

$$d(z, v_i) + d(x, z) - (d(z, v_k) - d(x, z)) > \epsilon$$

and therefore since $d(z, v_i) = d(z, v_k)$ we have that

$$d(x, z) > \frac{\epsilon}{2},$$

which is equivalent to

$$d(x, \mathcal{V}_i) > \frac{\epsilon}{2}.$$

■

C. Proof of the Theorem 3

To prove the theorem, we first introduce the following lemma.

Lemma 1: Let Ψ be the space transformation given by the following mapping: $\Psi : (\mathcal{D}, d) \to (\mathbb{R}^m, L_\infty)$. Where

$$\Psi(x) \triangleq (d(x, b_1), \ldots, d(x, b_m)).$$

For which, we can bound the distance $d(x, y)$

$$L_\infty(\Psi(x), \Psi(y)) \leq d(x, y),$$

where

$$L_\infty(a, b) \triangleq \max_{j=1}^{m} |a_j - b_j|.$$

Then we can straightforwardly bound the distance between objects and subsets

$$L_\infty(\Psi(x), \mathcal{A}) = \max_{j=1}^{m} |d(b_j, x) - d(b_j, \mathcal{A})| \leq d(x, \mathcal{A}). \tag{19}$$

Proof: Since the triangle inequality holds also for the distance between sets and objects [27], we can write

$$d(x, \mathcal{A}) \geq |d(b_j, x) - d(b_j, \mathcal{A})| \quad \forall j$$

and hence

$$d(x, \mathcal{A}) \geq \max_{j=1}^{m} |d(b_j, x) - d(b_j, \mathcal{A})| = L_\infty(\Psi(x), \mathcal{A}))$$

■

We can now prove the main Theorem 3.

Proof:

In order to prove this theorem, we must verify that $\mathbf{B}_N^\epsilon$ includes the general ϵ-Cover of $\mathbf{B}_N$, i.e., that $\mathbf{B}_N^\epsilon \supseteq O^\epsilon(\mathbf{B}_N)$ is always true, which can be translated in the following condition:

$$x \in \mathcal{P}_i^\epsilon \Rightarrow x \in \mathcal{B}_i^\epsilon, \tag{20}$$

where $\mathcal{P}_i^\epsilon \in O^\epsilon(\mathbf{B}_N)$ is defined as in (1), i.e.,

$$\mathcal{P}_i^\epsilon \triangleq \{x \in \mathcal{D} \mid d(x, \mathcal{B}_i) \leq \frac{\epsilon}{2}\} \tag{21}$$

From (9) condition (20) can be expressed as

$$d(x, \mathcal{B}_i) \leq \frac{\epsilon}{2} \Rightarrow \exists k \neq i \mid x \in \mathcal{B}^{\epsilon}_{ik} \vee x \in \mathcal{B}_i,$$

which can be also written as:

$$x \notin \mathcal{B}_i \wedge x \notin \mathcal{B}^{\epsilon}_{ik} \forall k \neq i \Rightarrow \ d(x, \mathcal{B}_i) > \frac{\epsilon}{2}.$$

When $x \notin \mathcal{B}_i$ and $x \notin \mathcal{B}^{\epsilon}_{ik}$ from (10) we have that there must exist a j for which either

$$d(x, b^i_j) > u^i_j + \epsilon/2, \tag{22}$$

or

$$d(x, b^i_j) < l^i_j - \epsilon/2, \tag{23}$$

is true. Let us consider the case the case in (22). Realizing that $u^i_j = d(b^i_j, \mathcal{B}_i)$, we can state that $d(x, b^i_j) - u^i_j = d(x, b^i_j) - d(b^i_j, \mathcal{B}_i) > \epsilon/2$ holds. This last inequality can be also written as

$$L_\infty(\Psi(x), B_i) > \epsilon/2.$$

However, because of (19), we can conclude

$$d(x, B_i) > \frac{\epsilon}{2}.$$

Analogously, it is possible to prove that inequality (23) leads to the same conclusion. ■

D. Proof of the Theorem 4

Proof: We prove this theorem by contradiction, by supposing that exits $i \neq j$ such that $\langle x, y \rangle \in \mathit{SSJ}^*_\epsilon(X_i)$ and $\langle x, y \rangle \in \mathit{SSJ}^*_\epsilon(X_j)$. It easy to see from (13) that $r_i(x, y) =$ ***true***, and $r_j(x, y) =$ ***true*** both hold. However, from (12), we have that $\min \sigma(x) \bigcap \sigma(y) = i$ and $\min \sigma(x) \bigcap \sigma(y) = j$ that implies that $i = j$, which is in contradiction with the initial assumption $i \neq j$. ■