CP-Index: Using Clustering and Pivots for Indexing Non-Metric Spaces

Victor Sepulveda
PRISMA Research Group
Department of Computer Science
University of Chile
vsepulve@dcc.uchile.cl

Benjamin Bustos
PRISMA Research Group
Department of Computer Science
University of Chile
bebustos@dcc.uchile.cl

ABSTRACT

Most multimedia information retrieval systems use an indexing scheme to speed up similarity search. The index aims to discard large portions of the data collection at query time. Generally, these approaches use the triangular inequality to discard elements or groups of elements, thus requiring that the comparison distance satisfies the metric postulates. However, recent research shows that, for some applications, it is appropriate to use a non-metric distance, which can give more accurate judgments about the similarity of two objects. In such cases, the lack of the triangle inequality makes impossible to use the traditional approaches for indexing. In this paper we introduce the CP-index, a new approximate indexing technique for non-metric spaces that combines clustering and pivots. The index dynamically adapts to the conditions of the non-metric space using pivots when the fraction of triplets that break the triangle inequality is small, but sequentially searching the most promising candidates when the pivots becomes ineffective.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

Keywords

Similarity Search, Non-Metric Distance, Indexing, Clustering, Pivots.

1. INTRODUCTION

Nowadays, the amount of multimedia information available is growing at an increasing rate, and systems to manipulate and search for information in multimedia repositories become increasingly important. As technology improves, the ability to generate multimedia objects, as well as its complexity, increases. Indeed, it is estimated that about 95% of the web space (measured in bytes) consists of multimedia information [11]. Applications in this area include: multimedia information retrieval systems, biometrics, databases for medicine, geology, astronomy, and many others.

A main problem in these applications is to perform similarity search in a database, ie, finding the objects most similar to a query. However, when the objects of the database are complex, comparing their similarity can be computationally expensive, so it is desired to avoid as many comparisons of objects as possible. The traditional approach is to use metric distances for comparison, so the triangle inequality can be used to prune the search [12]. However, there are many cases where it is more appropriate to use distances that do not satisfy the metric properties, the so-called non-metric spaces.

There are a variety of reasons that justify the use of non-metric distances, eg, robustness to outliers, the use of machine learning models to make better comparisons, and modeling of human perception [11, 10, 7]. One of the most important reasons is to emulate the perceptual similarity. Certain studies show that human perception have no metric behavior [10, 7], since it do not uses a static set of criteria to judge similarity. Instead, it determines the most appropriate criteria for a particular comparison, and then use them to evaluate similarity. Thus, different pairs of objects are compared with different measures, and properties like the triangle inequality lose their meaning.

The triangle inequality is the base of most indexing techniques, so it is interesting to develop techniques to deal with the indexing in non-metric spaces. It is also desirable that the techniques are not dependent on a particular space, but based only on the pairwise distances and the information derived from them.

In general, the use of pivots is not considered in spaces that fail in the triangle inequality because it leads to false dismissals. However, in this paper we study the effect of clustering the data on the fails caused by pivots. We introduce the CP-index, a new approximate indexing technique that combines clustering and pivots for non-metric indexing. This technique adapts to the conditions of non-metric space, using pivots when the fraction of triplets that break the triangle inequality is small, but sequentially searching the most promising candidates when the use of pivots becomes useless, in a dynamic way. We experimentally tested this strategy with the L_p Fractional distance to represent spaces in which the triangle inequality is broken in different degrees.

The rest of this paper is organized as follows: in Section 2 we introduce the problem of similarity search, indexing in

Table 1: Notation Used

Symbol	Meaning
$\mathbb{U}$	Universe (descriptors of multimedia objects)
$\mathbb{S}$	Database of objects ($\mathbb{S} \subset \mathbb{U}$)
O_i, O_j	Database Objects ($O_i, O_j \in \mathbb{S}$)
Q	Query Object ($Q \in \mathbb{U}$)
$\sigma : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{R}$	Similarity function
$\delta : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{R}$	Distance function

metric and non-metric spaces, and give some examples of non-metric distances measures. In Section 3 we explain the main approaches proposed so far for indexing non-metric spaces. In Section 4 we introduce our proposal, the CP-Index. In Section 5 we present experimental results, and in Section 6 we conclude and present future work.

2. SIMILARITY SEARCH

In modern multimedia retrieval systems, the similarity search process has become its central task. Therefore, improving its efficiency and efficacy has become very important, since the complexity of multimedia objects and the volume of multimedia repositories grows up quickly.

First of all, we consider the similarity model. Table 1 shows the notation used in this paper. We have a distance function that measures the dis-similarity of two database objects as follows:

$$\delta(Q, O_i) < \delta(Q, O_j) \Longleftrightarrow \sigma(Q, O_i) > \sigma(Q, O_j).$$

This means that Q is more similar to O_i than to O_j. In the rest of this paper, The principal query modalities, range search and knn search, commonly used in multimedia retrieval systems, are defined as follows:

Definition 1. Let $Q \in \mathbb{U}$. The range query $range(Q, r)$, with tolerance radius $r \in \mathbb{R}^+$, returns the set:

$$Range = \{O_i \in \mathbb{S} \mid \delta(O_i, Q) \leq r\}$$

Definition 2. Let $Q \in \mathbb{U}$. The query for the K nearest neighbors $Knn(Q)$, with $K \in \mathbb{N}^+$, returns a set that satisfies the following:

$$|Knn| = K,\ \forall O_i \in Knn,\ \forall O_j \in \mathbb{S}{-}Knn,\ \delta(O_i, Q) \leq \delta(O_j, Q)$$

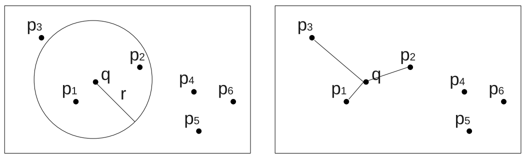

Figure 1: Example range query (left) and 3nn query (right). The first report the objects P_1 y P_2 and the second, the elements P_1, P_2 y P_3.

2.1 Metric Spaces

Traditionally, a metric distance function has been used as dis-similarity measure. The rationale behind this is to exploit the topological characteristics of a metric distance to facilitate indexing and to speed up the searches. Now we review the metric distance properties.

Definition 3. A distance function $\delta : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{R}$ is metric if and only if the following properties are satisfied:

Reflexivity	$\delta(O_i, O_j) = 0 \Longleftrightarrow O_i = O_j$
Non-negativity	$\delta(O_i, O_j) > 0 \Longleftrightarrow O_i \neq O_j$
Symmetry	$\delta(O_i, O_j) = \delta(O_j, O_i)$
Triangle Inequality	$\delta(O_i, O_j) + \delta(O_j, O_k) \geq \delta(O_i, O_k)$

2.2 Indexing

Generally, a distance function that appropriately models similarity between complex objects can be computationally expensive. Therefore, it is desirable to avoid as many distance calculations as possible while performing a similarity search. The traditional approach is to index the space or to classify the elements in some way to discard entire zones or groups of objects. The fundamental property for indexing the database and speed up the search is the triangle inequality, and most indexing techniques are based on it.

The different indexing strategies can be divided into two main approaches: partitioning the space or using pivots. The former divides the space into zones characterized by some of the objects contained in them, and the query object is compared with the elements that represent each zone, trying to discard entire zones if they are far enough. The latter uses a set of objects as pivots. The distances between these pivots and the rest of the database are computed and stored. At query time, the stored distances and the distances between the query and the pivots are used to compute lower bounds to the distance between query and objects, allowing the algorithm to discard non relevant objects.

2.3 Non-Metric Spaces

Any distance function that fails to satisfy any of the metric properties is a non-metric distance. There are a variety of reasons that justify using non-metric distances, like robustness to outliers, using machine learning models whose behavior can be difficult to predict, using black box distances whose exact behavior is unknown, or a better modeling of the human perception. Furthermore, restricting the domain expert to generate a similarity measure satisfying the metric postulates may be harmful to the effectiveness of retrieval, or even impossible.

Among the metric properties, failures in satisfying reflexivity or non-negativity (by distances that allow certain objects to be differently self-similar) and in symmetry (for example, if the similarity measure allows an object containing another one to be more different to it than vice versa) can be repaired relatively easily (e.g., adding a constant value to the distance). Distances that do not satisfy the triangle inequality tend to be much more difficult to treat, as this particularly affects the indexing techniques. Distance functions that satisfy the metric properties except the triangle inequality are called semi-metric and are of special interest. For the remainder of this paper, we will consider only the case of semi-metrics when referring to non-metric distances.

2.4 Examples of Non-metric Distances

We now present some relevant examples of distances that do not satisfy the metric properties, in particular, the triangle inequality. In addition to these and several other, any combination of non-metric distances can lead to a non-metric, and even certain combinations of metric distances may lead to non-metric distance measures.

2.4.1 *Dynamic Partial Function*

Dynamic Partial Distance Function [6] is based on the family of Minkowski distances, but uses only a few coordinates to calculate dis-similarity. The idea is to consider only a fraction of the most similar coordinates, assuming that objects that should be classified as "similar" may have a small number of very different components. DPF distance is defined as follows:

Let $c_i = |x_i - y_i|$ where x_i, y_i are the i-th coordinate of x and y. Now let Δ_m be the set of the m smallest c_i from $\{c_1, ...c_d\}$.

$$DPF_p(x,y) = \left(\sum_{c_i \in \Delta_m} {c_i}^p \right)^{1/p}$$

2.4.2 *Cosine Distance*

Cosine distance [2] measures the difference in the direction of two vectors, regardless of their magnitude. It is commonly used in text and documents retrieval, and is defined as follows:

$$\sigma_{cos}(x,y) = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2 \sum_{i=1}^{d} y_i^2}}$$

σ_{cos} is a measure of similarity, but it can be made a distance by defining $\delta_{cos} = 1 - \sigma_{cos}$, which is a non metric distance function.

2.4.3 *Dynamic Time Warping*

Dynamic Time Warping [3] is a measure commonly used to compare the distance in time series. Let $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ be two time series, and M a $n \times m$ matrix with $M[i,j] = (x_i - y_j)^2$. A warping path $W = \{w_1, ...w_t\}$ is a list of components of M that satisfy the following properties: $w_1 = M[1,1]$ and $w_t = M[m,n]$, for $w_k = M[a,b]$ and $w_{k-1} = M[a', b']$ then $a - a' \leq 1$ and $b - b' \leq 1$, and for $w_k = M[a,b]$ and $w_{k-1} = M[a', b']$ then $a - a' \geq 0$ and $b - b' \geq 0$. DTW looks for the minimum warping path, so the distance is defined as follows:

$$DTW(x,y) = min_W \left\{ \sqrt{\sum_{k=1}^{t} w_k} \right\}$$

DTW fails to satisfy the triangle inequality so it is a semi-metric.

2.4.4 *Fractional L_p Distances*

The Minkowski family are perhaps the most widely used distances as a measure of dis-similarity in vector spaces. Minkowski distance with parameter p, L_p, is defined as follows:

Let $x, y \in \mathbb{R}^d$ be two vectors of dimension d and, $p \geq 1$.

$$L_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p}$$

The fractional L_p distance [1] extends this concept by allowing a parameter $0 < p < 1$, but that results in a semi metric distance function, because it no longer satisfy the triangle inequality. As p gets close to 0, the space generated by the fractional L_p distance becomes "less metric", increasing the fraction of triplets that break the triangle inequality. Below is a graph showing the percentage of failure of the triangle inequality in a set of uniform distributed vectors in the unitary cube of dimension 8 for different values of p.

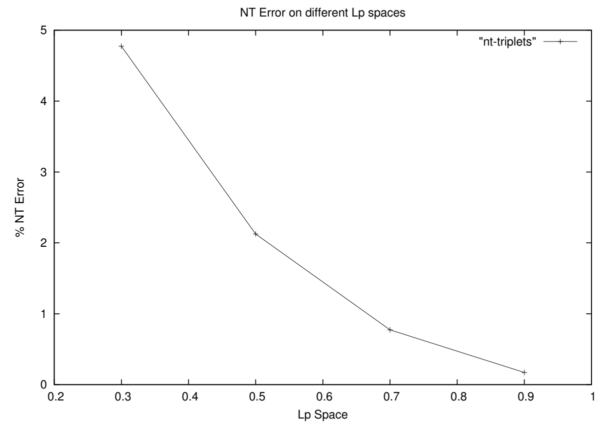

Figure 2: Failure rate of the Triangle Inequality for different fractional L_p norms.

This distance has the advantage of allowing us to control the rupture of the triangle inequality by changing the parameter p. Similar to the dimension in a synthetic vector space, with the Fractional L_p distance we can make tests on "more non-metric" spaces, where the fraction of triplets that break the triangle inequality is high, or closer to a metric in which the rate of the triangle inequality failure is low.

3. NON-METRIC INDEXING METHODS

Although there are indexing techniques for particular cases of non-metric spaces, such as the inverted index for documents, it is of interest to develop techniques that can deal efficiently the similarity search problem in generic non-metric spaces. Here, the pairwise distances are the only information used for indexing and querying. In the following, we review the main approaches proposed so far to address this problem.

3.1 Constant Shifting Embedding

There are simple ways to convert a semi-metric δ into a metric distance. One way is to add a suitable constant to the distance, to obtain a modified distance $\delta' = \delta + c$. The triangle inequality can be enforced with a large enough constant [9]. The problems of this approach are that the appropriate constant should be calculated considering all the data if there is no adequate analytical information, otherwise it might not be large enough for a new query object. A more difficult problem is that, in general, the appropriate constant to turn the semi-metric into a metric is too big and the new space becomes hard to index, because all distances become relatively too similar and the triangle inequality can no longer discard elements.

A more elaborated approach is the recently introduced Local Constant Embedding [4], where the space is divided into groups and the method looks for a suitable constant for each group. This allows the index to use smaller constants, permitting a better indexing of each group as an independent metric space. However, the choice of appropriate groups is complex and computationally expensive: the proposed heuristic requires the whole distance matrix and it is cubic (in time) in the number of elements.

3.2 QIC

QIC [5] is an approach based on Filter&Refine, using several distance measures. Its application in non-metric spaces requires the use of a metric for building the index, that is a lower bound of the non-metric query distance. The search algorithm traverses the index using the lower bound distance, and then refines the result with the non-metric distance. This technique is implemented by modifying an M-tree to generate the so-called QIC-M-Tree. The disadvantage of this method is that finding a distance metric that is a lower bound of a complex non-metric can be difficult (or even impossible for black-box distances), and it also must be a tight lower bound for the search to be reasonably efficient.

3.3 TriGen

The TriGen algorithm [11] tries to modify the semi-metric by applying a concave or convex function to make it more or less metrical (reducing or increasing the triangular inequality error). In this way, the space becomes "less indexable" as it becomes more metric. The generation of the modifier function is performed automatically and the modified distance, turned into an almost metric, can be used in a metric access method for an approximate search.

3.4 Classification Techniques

Another approach that has been used for searching in non-metric spaces is to address the problem of similarity search as a classification problem. This approach uses statistical information or performs clustering of the data in some way to generate a set of classes or clusters. At query time, the query object is classified into one or more classes, and the corresponding clusters are searched for objects, assuming that the most similar objects to the query will be found in the most promising clusters.

An indexing technique using this approach is DynDex [6], which performs clustering with the CLARANS [8] algorithm in the construction phase. At query time, DynDex sequentially scans a fixed number of the most promising clusters.

4. CP-INDEX

4.1 Using Pivots in Non-Metric Spaces

Using pivots to speed up the search in a space where the triangle inequality fails may lead to false dismissals. As shown in the following example on the $L_{0.5}$ space, the commonly used method to compute a lower bound of the actual distance between a candidate and an query object fails. That is, the lower bound distance obtained using the pivot is not valid. Thus, the object would be incorrectly discarded and lost for the final result.

Figure 3 shows the example in 2-D and $L_{0.5}$ distance. There is an object p pivot for a candidate object x. The $L_{0.5}$ distance between the pivot and the candidate is 2. We have a query object q at an approximate distance of 5.82 of the pivot, and a query radius of 1. In this situation, the candidate would be dismissed because $|5.82 - 2| > 1$ (the lower bound distance is computed as $|\delta(p, x) - \delta(p, q)|$, see Zezula et al. [12] for details). But, the distance between the query object and the candidate is 1, thus it was relevant and incorrectly discarded.

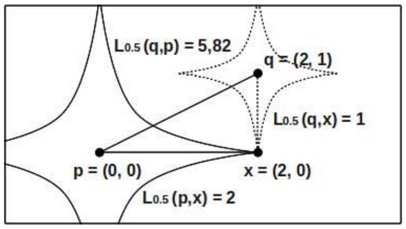

Figure 3: Example of a false dismissal on the 2-D and $L_{0.5}$ space.

However, there are cases in which this event is less frequent, depending on the chosen pivots and the relationship between the query object and the candidate to be discarded. Indeed, we experimentally discovered that when the database objects are separated into compact groups of close objects and the pivots are locally chosen and used at each group, less objects are incorrectly discarded than when using the same pivots but globally, for the whole database. Indeed, we conducted tests where we generated clusters of different qualities, and we studied the effectiveness of pivots in them. We partition the data into clusters and choose at random a certain percentage of elements as pivots. We used a set of random query objects, counting the number of times the triangle inequality fails (that is, the number of times a candidate is discarded incorrectly). We studied the effect of increasing the number of groups from one (i.e., the entire database), to groups with a small number of objects.

We observed that the number of false dismissals is reduced by having more clusters, because they are more compact. This was achieved by partitioning the database into clusters, by simply choosing centers at random, and by randomly choosing pivots in each cluster. Figure 4 shows the average number of false dismissals depending on the number of such clusters in a database of 10,000 vectors of dimension 112 in the $L_{0.5}$ space. Therefore, local pivots may be used for indexing some non-metric spaces, but it is not guaranteed that all relevant objects will be found.

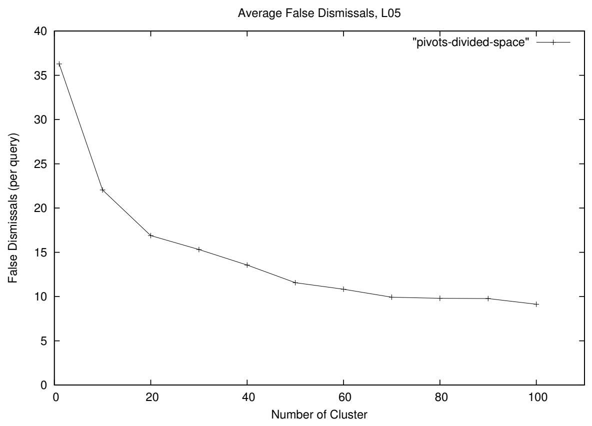

Figure 4: Reduction of false dismissals on the $L_{0.5}$ space by clustering the data.

4.2 Clustering&Pivots Index

The false dismissals arise from the relationship between the candidate to be discarded, the query object, and the pivots, that generate the breaking of the triangle inequality. However, there are spaces in which the triangle inequality fails too often. A better pivot selection technique can reduce failures, but testing in spaces where the failure rate of the triangle inequality is greater, such as the $L_{0.3}$, showed its inefficacy. In such spaces, most techniques that were tested fail or turn into sequential search. We then prepare a technique that uses pivots to reduce the work, but adapting to the characteristics of non-metric space, linear scanning when the pivots fails.

CP-Index uses a similar approach to that of DynDex, and considers the similarity search as a classification problem. During the construction phase a clustering process is performed using CLARANS, and in each cluster a set of pivots is chosen with some heuristics. Subsequently, the elements of each cluster are classified according to whether they can be correctly discarded by pivots or not, with the chosen pivots and a representative set of queries. At query time, we try to discard by pivots only those elements which have been classified as treatable by pivots. For the remaining elements, we consider that the pivots test will fail and calculate its actual distance to the query. In this way, the index adjusts and uses pivots only when it is safe. In spaces where the triangle inequality fails too often, such as $L_{0.3}$, sequentially searches the clusters while in more metric spaces, such as the $L_{0.7}$, uses pivots to discard candidates quickly. Finally, we set the amount of work (measured in distance computations) to be performed during the search. Thus, the CP-Index implements an approximate similarity search.

The CLARANS method is a clustering technique that uses only the pairwise distances. It does not depends on metric or vectorial properties of the space, but it does receives the number of clusters to generate. It relies on random search of centers that tend to improve the quality of clustering.

To explain the CLARANS method, we can use an analogy between the clustering process and the search in a graph. Suppose a graph $G_{n,k}$ in which each node represents a particular clustering. The Node N_m is formed by the set of k centroids $\{O_{m_1}...O_{m_k}\}$. Two nodes are neighbors (they are connected by an arc) if their sets of centroids differ by a single element. Also, each node has a cost equal to the sum of the distances of each element to the centroid of its cluster. This is a simple way of measuring the inverse of the quality of that particular clustering.

The method searches the graph for a node that minimizes the cost. However, instead of thoroughly searching each neighbor of each node, it starts from any node and looks only a few randomly chosen neighbors to continue. CLARANS stops when it has found enough neighbors of the current node that does not improve its quality. At this point, it assumes that the current node is good enough to be considered a local minimum in the graph. Then the process is repeated starting from a different node to find a certain number of local minimum and it keeps the best of them.

To test whether an object can be discarded by pivots, we test it with a set of queries and the pivots of its cluster. If any query leads to a false dismissal with those pivots, the object is marked as non treatable by pivots. We could make a more elaborate process, in which it would be necessary a certain number pivots that erroneously discard the element before marking it as untreatable, or which seeks to improve the set of pivots when the tests fail, but the simple test-and-mark process performed better than several heuristics.

The query set used in the test is important. If we had the future query objects, the test would mark as treatable by pivots only the elements that will not be erroneously discarded. However, if we assume that the query objects will have the same distances distribution of the elements in the database, it would suffice to choose appropriate elements of the same database for querying. Moreover, if the clustering process we have done was effective, then the representatives of the clusters will be good substitutes for the elements of the cluster. Therefore, we use only the representatives of the clusters as query objects to test and mark the database elements.

Here we present the pseudocode of the main algorithms involved in construction. The construction method (Algorithm 1) receives the dataset, the number of clusters and distance to be used. It makes the clustering of the data using the method MAKE_CLUSTERING, that can use the most appropriate algorithm for this. We use the CLARANS [8] algorithm, allowing it an appropriate amount of work to achieve a good trade-off between construction time and query efficiency. Then we use the method CHOOSE_QUERIES to choose a representative set of queries from the different clusters. In our tests we used the representatives of the clusters as query objects. Then, for each cluster we choose a set of pivots with some heuristic, and verify them with the method VERIFY_PIVOTS. In the tests we choose the pivots randomly, and the verification process is described in Algorithm 2. This method verifies that the triangle inequality is satisfied for the queries and the chosen pivots, for each element of the cluster. If the verification fails for any element of the cluster with any query, the element is marked as non-treatable by pivots. At query time, only the elements that have not been marked as intractable will be tested with the pivots. This test system allows that the decision of whether using pivots or not is taken dynamically depending on the characteristics of the space. If more triplets do not satisfy the triangle inequality, more elements will be directly checked against the query.

Algorithm 1 CONSTRUCTION

```
Require: Dataset $\mathbb{S}$, number of clusters $K$, Distance $\delta$
1: Let $\mathbb{C}$ be a set to store the clusters
2: MAKE_CLUSTERING($\mathbb{S}, K, \delta, \mathbb{C}$)
3: Let $\mathbb{Q}$ be a set to store the queries
4: CHOOSE_QUERIES($\mathbb{C}, \mathbb{Q}$)
5: for Each Cluster $C$ of $\mathbb{C}$ do
6:     PREPARE_PIVOTS($C, \delta$)
7:     VERIFY_PIVOTS($C, \delta, \mathbb{Q}$)
8: end for
```

5. EXPERIMENTAL EVALUATION

5.1 Experimental Setting

For the tests we used a database of 112.682 color histograms of images of 112 components from the SISAP Library[1]. We used the distances $L_{0.3}$, $L_{0.5}$, $L_{0.7}$ and $L_{0.9}$ to

[1]http://sisap.org/Home.html

Algorithm 2 VERIFY_PIVOTS

Require: Cluster C, Distance δ, Query Set $\mathbb{Q}$

```
 1: for Each Query $Q$ of $\mathbb{Q}$ do
 2:   for Each Pivot $P$ of $C$ do
 3:     Compute $\delta(Q,P)$
 4:   end for
 5:   for Each Object $X$ of $C$ do
 6:     Compute $\delta(Q,X)$
 7:     for Each Pivot $P$ of $C$ do
 8:       if $|\delta(Q,P)-\delta(P,X)|>\delta(Q,X)$ then
 9:         Mark $X$ as non-treatable by pivots
10:         break
11:       end if
12:     end for
13:   end for
14: end for
```

test the behavior of the technique in different spaces with different failure rates in the triangle inequality.

We also run tests in the space of Normalized Edit Distance with a English dictionary of 69069 words from the SISAP Library. Another experiment was conducted with a database of 12218 time series of 962 components, using the DTW distance.

We randomly choose 10% of each database as query objects, and the rest was used to build the indexes. For each query object, we linearly search for their 10 nearest neighbors in advance, and stored those results. Then, we performed the clustering process to index databases. We search in each index, allowing different amounts of work (measured in number of distance calculations), and verified the percentage of correct answers obtained on average for each of the 10NN queries. In each space we tested three versions of the index: one with 0 pivots (in a similar way to DynDex, but with limited work in place of a fixed number of clusters to search), one with pivots but without the test that verifies the feasibility of the pivots, and one with pivots and the verification test.

CLARANS clustering method divides the data set into a fixed number of clusters, and its cost is approximately $O(KNW)$ where N is the number of elements, K is the number of clusters, and W is a factor that determines the allowed work of the algorithm.

The parameter W indicates the number of neighbors of each node to be visited during the search of a good enough clustering (as described in section 4.2) and can be $O(1)$, $O(log(N))$, $O(N)$, or any other, which changes the quality of the result. In the tests we used $log(N)$, for a good trade-off between construction time and result. At query time, the index was allowed to make a number of distance calculations equal to 2% of the database, 4%, 6%, 8% and 10%. In each case, the search process is terminated when it reaches the quota of allowed work. Below are the results comparing the three versions.

The number of clusters is also important for construction time and query efficiency. If we assume that we will visit a fixed number of c clusters, and consider the average size of a cluster, then we can see that the number of distance calculations performed during the query will be:

$$Cost_{dist} = K + \frac{cN}{K}$$

The first part is to classify the query in one or more clusters (because there are K centroids), and the second is the cost of sequentially searching (in the worst case) each of them. If we seek for the optimal number of clusters, we obtain $K = O(\sqrt{N})$ clusters.

$$Cost'_{dist} = 1 - \frac{cN}{K^2} = 0 \Longrightarrow cN = K^2 \Longrightarrow K^{opt} = \sqrt{cN}.$$

However, that number of clusters is valid only for the case of linear scanning. It is difficult to find an optimal number of clusters considering the extra work required to use pivots, and the distance calculations saved by them. Still, we experimentally found that $K = \sqrt{N}$ clusters gives a good trade-off between construction time and efficiency for the cases tested. In practice, it may be necessary to perform some tests to find the best number of clusters for a particular instance.

5.2 Results

Figure 5 shows that in space $L_{0.3}$ the technique with verified pivots behaves almost indistinguishable to the linear scan of the most promising clusters. That is because in that space the triangle inequality fails too often, and the pivots test fails in most cases, and only a few objects can be checked to save distance calculating. It can be see that using pivots without verification in this space leads to no more than 42% of correct results, even with the maximum allowed work.

In the spaces $L_{0.5}$ and $L_{0.7}$ (Figure 6 and Figure 7), the technique with verified pivots performs better than the linear scanning, although using pivots without verification can improve the result in some ranges of allowed amount of work. That is, with few distance computations it gets a reasonable result, but it is unable to obtain higher percentages of correct answers because many objects were wrongly discarded.

In the space $L_{0.9}$ (Figure 8), using unverified pivots works better, because the space is almost metric. However, the technique with verified pivots works almost as good as the unverified technique, showing its adaptability and giving results almost identical (or even better) to the best of the other two versions in each space.

Figure 9 shows that in the space of Normalized Edit Distance, the technique with verified pivots and the unverified version behaves in a very similar way, a little better than the linear scanning. This can also be seen in Figure 10, for the space DTW.

So, CP-Index is able to dynamically adapt to the space, taking advantage of the pivots only when it is safe.

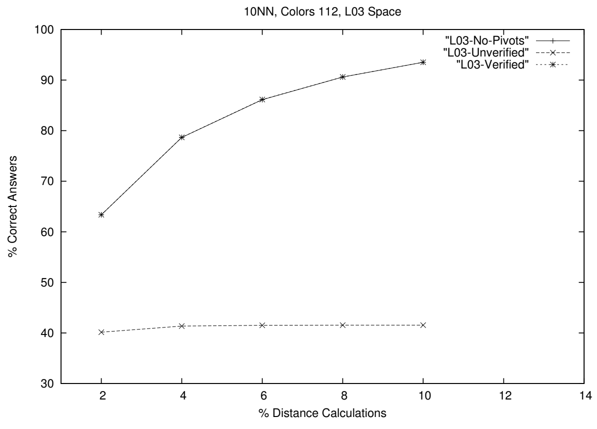

Figure 5: KNN in $L_{0.3}$ Space

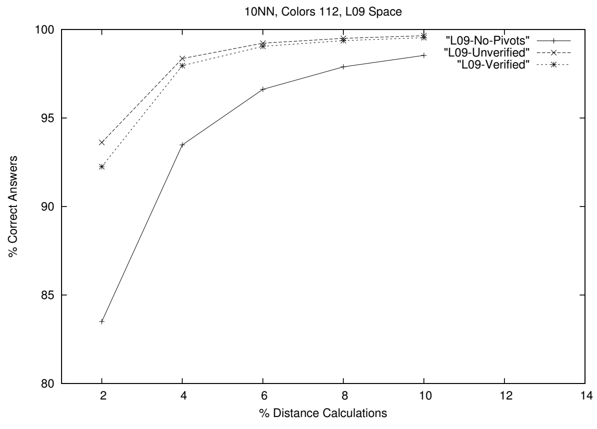

Figure 8: KNN in $L_{0.9}$ Space

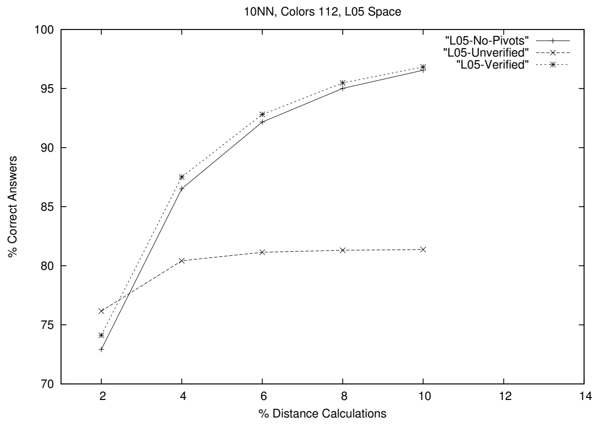

Figure 6: KNN in $L_{0.5}$ Space

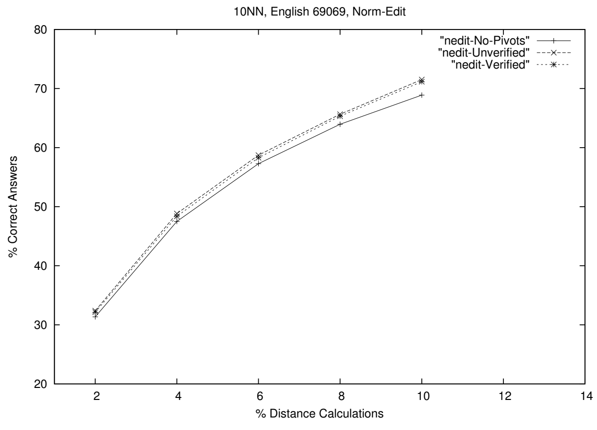

Figure 9: KNN in Normalized Edit Distance Space

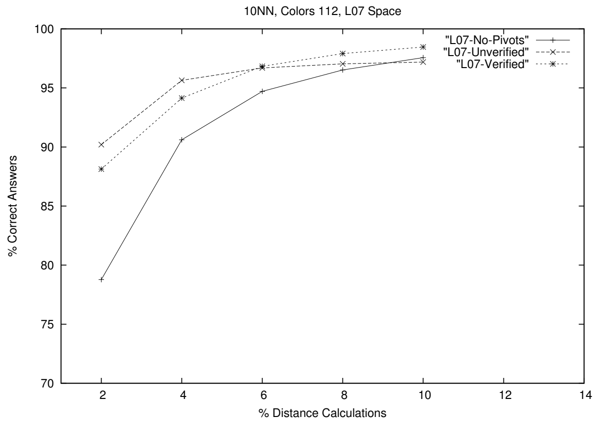

Figure 7: KNN in $L_{0.7}$ Space

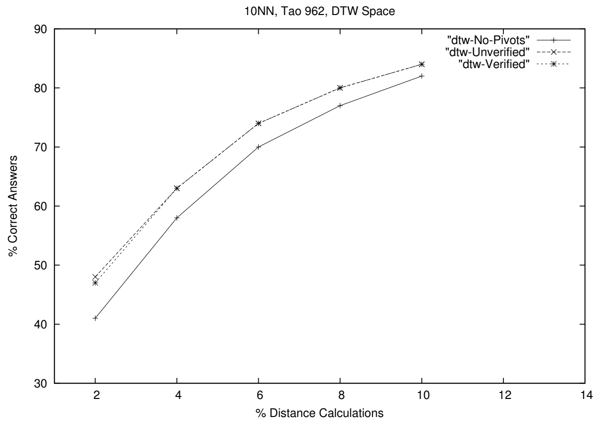

Figure 10: KNN in DTW Space

6. CONCLUSIONS

We have proposed a new approximate technique for indexing non-metric spaces which combines clustering and pivots. CP-Index adapts to the conditions of the non-metric space and tries using pivots, whenever it is possible and safe. However, when the breaking of the triangle inequality is large, it dynamically avoids using pivots preferring a sequential search of the most promising candidates. This combination allows the index to behave quite efficient in almost metric spaces, but it is capable of handling in a reasonable way most difficult spaces in which the triangle inequality fails often.

Future work, currently in development, focuses on improving the technique and its experimental comparison. On the one hand, it is under development a version that allows a finer selection of promising clusters, smaller but in greater number, to get a better response. We are also working on a dynamic version that allows insertions and deletions. On the other hand, we are working on the experimental comparison of these proposals with the major indexing techniques proposed so far for non-metric spaces.

7. REFERENCES

[1] C. Aggarwal, A. Hinneburg, and D. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In *Proc. 8th International Conference on Database Theory (ICDT'01)*, London, UK, 2001. Springer-Verlag.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[3] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *Proc. AAAI Workshop On Knowledge Discovery in Databases*, pages 229–248, 1994.

[4] L. Chen and X. Lian. Efficient similarity search in nonmetric spaces with local constant embedding. *IEEE Transactions on Knowledge and Data Engineering*, 20(3):321–336, 2008.

[5] P. Ciaccia and M. Patella. Searching in metric spaces with user-defined and approximate distances. *ACM Database Systems*, 27(4):398–437, 2002.

[6] K. S. Goh, B. Li, and E. Chang. DynDex: A dynamic and non-metric space indexer. In *Proc. 10th ACM International Conference on Multimedia (MM'01)*, pages 466–475. ACM Press, 2002.

[7] D. Jacobs, D. Weinshall, and Y. Gdalyahu. Classification with nonmetric distances: Image retrieval and class representation. *IEEE Pattern Analysis and Machine Intelligence*, 22(6):583–600, 2000.

[8] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th VLDB*, pages 144–155, 1994.

[9] V. Roth, J. Laub, J. M. Buhmann, and K. R. Müller. Going metric: Denoising pairwise data. In *Proc. International Conference on Neural Information Processing Systems (NIPS'02)*, pages 817–824, 2002.

[10] S. Santini and R. Jain. Similarity measures. *IEEE Pattern Analysis and Machine Intelligence*, 21(9):871–883, 1999.

[11] T. Skopal. Unified framework for fast exact and approximate search in dissimilarity spaces. *ACM Transactions on Database Systems*, 32(4):1–46, 2007.

[12] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search: The Metric Space Approach (Advances in Database Systems)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.