Pattern Recognition Letters 15 (1994) 9–17
North-Holland

January 1994

PATREC 1140

A new version of the Nearest-Neighbour Approximating and Eliminating Search Algorithm (AESA) with linear preprocessing time and memory requirements

María Luisa Micó, José Oncina

Universidad de Alicante, Departamento de Sistemas Informáticos y Computación, Alicante, Spain

Enrique Vidal

Universidad Politécnica de Valencia, Departamento de Sistemas Informáticos y Computación, Valencia, Spain

Received 13 April 1992

Abstract

Micó, M.L., J. Oncina and E. Vidal, A new version of the Nearest-Neighbour Approximating and Eliminating Search Algorithm (AESA) with linear preprocessing time and memory requirements, Pattern Recognition Letters 15 (1994) 9–17.

The Approximating and Eliminating Search Algorithm (AESA) can currently be considered as one of the most efficient procedures for finding Nearest Neighbours in Metric Spaces where distance computation is expensive. One of the major bottlenecks of the AESA, however, is its quadratic preprocessing time and memory space requirements which, in practice, can severely limit the applicability of the algorithm for large sets of data. In this paper a new version of the AESA is introduced which only requires linear preprocessing time and memory. The performance of the new version, referred to as 'Linear AESA' (LAESA), is studied through a number of simulation experiments in abstract metric spaces. The results show that LAESA achieves a search performance similar to that of the AESA, while definitely overcoming the quadratic costs bottleneck.

Keywords. Metric spaces, triangle inequality, fast nearest-neighbours searching algorithms, pattern recognition.

1. Introduction

Nearest-Neighbours (NN) techniques have become increasingly popular in Pattern Recognition (Fukunaga (1990), Dasarathy (1991)). These techniques become especially important if no reasonable vector space exists where the objects or points can be adequately represented, though a convenient procedure is available for computing an appropriate *dissimilarity measure* or *distance* between every pair of points. In this case, there are many problems of

Correspondence to: M.L. Micó, Universidad de Alicante, Departamento di Sistemas Informáticos y Computación, Alicante, Spain.
Work partially supported by the Spanish CICYT grant TIC-448/89.

(practical) interest in which the distance computation is particularly expensive. (Isolated) Word Recognition through Dynamic Time Warping (Rabiner and Levinson (1984), Casacuberta and Vidal (1987)), Attributed Graph Match Searching (Sanfeliu and Fu (1983), Shapiro and Haralik (1985)) or best-match String Edit searching (Marzal and Vidal (1992)), to name but a few, are examples of these problems. While Nearest-Neighbours techniques are perhaps the only available in these cases, the large costs of distance computing drastically limit the size of the problems that can be afforded. afforded.

In order to alleviate the computational burden in these cases many techniques for *fast* Nearest-Neighbour Searching have been proposed in the last few years (Dasarathy (1991)). Particularly efficient is the algorithm known as 'Approximating and Eliminating Search Algorithm' (AESA), which achieves NN search with an *average constant number of distance computations*; i.e., a number of distance computations that does not depend on the size of the set considered. This algorithm was introduced by Vidal (1986) and has been very successfully applied since then to moderately sized practical problems of Speech Recognition (Vidal et al. (1988), Vidal and Lloret (1988)).

Given a set P of *prototypes* and a test sample x, the AESA searches for a prototype in P which is a Nearest Neighbour of x through a best-first Branch and Bound strategy (Vidal (1994)). This strategy relies on a tight, Triangle-Inequality-based, lower bound function both for successively *selecting* candidate prototypes for distance computation (*'Approximating'*) and for *pruning-out* those prototypes with lower bound values greater than the best (smallest) distance found so far (*'Eliminating'*). The proposed lower bound function relies on a direct application of the Triangle Inequality that makes extensive use of previously computed distances. In particular, all distances between every pair of prototypes need be available during the search process. Obviously, this entails quadratic memory space requirements for storing these distances and a corresponding quadratic preprocessing time for their computation. Unfortunately, these quadratic costs severely limit the size of the problems the AESA can be practically applied to.

While very satisfactory solutions to this problem have been obtained by Ramasubramanian (1991) and Ramasubramanian and Paliwal (1990, 1992) for regular vector-space represented data, the problem remained open so far in the most general (and interesting) setting of Metric-Space data representations.

In this paper a new Metric-Space NN Search algorithm is introduced, which definitely overcomes the AESA quadratic bottleneck. This algorithm, called 'Linear AESA' (LAESA) (Micó et al. (1991)), only requires preprocessing time and memory space that grow linearly with the number of prototypes and achieves a search efficiency which is very close to that of the original AESA. It is based on similar (but independently developed) ideas as in Ramasubramanian and Paliwal (1990), but if offers the increased generality of not requiring the data to be represented in any vectorial form. The basic idea of LAESA is to attempt an AESA-like search for NN's, while relying only on the distances from a (small) subset of *'Base Prototypes'* to the remaining prototypes. For this purpose, LAESA requires a *preprocessing procedure* that not only computes the required distances, but also selects the corresponding Base Prototypes in linear time. The *search procedure*, on the other hand, consists of a direct extension of the best-first Branch and Bound formulation of the AESA (Vidal (1994)), in which only *a subset of interprototype distances* is available. This entails the need for an appropriate Base-Prototype management policy since, in principle, these prototypes are candidates for both selection (approximation) and pruning (elimination) like any other non-Base Prototype.

The proposed algorithms and experiments assessing their good performance will be presented in the following sections.

2. Selection of base prototypes

Since the search strategy will fully rely on the distances from all the prototypes to those selected as the set of Base-Prototypes (BP), making a good choice of this set is of particular concern. Obviously, search efficiency will depend not only on the amount of BPs selected, but also on their actual location with respect to the other prototypes. This last issue was already discussed in an early work by Shapiro (1977) with regard to a previous NN (best-match file) search algorithm (Burkhard and Keller (1973)). The results of this work suggest that certain improvements could be achieved by locating reference points *far away* from data clusters. Following such a suggestion, a greedy procedure can be proposed that attempts finding BPs which are *maximally separated.*

Starting with an arbitrarily selected BP, the proposed procedure computes the distances to all the remaining prototypes. The computed distances are restored in an array for their future use by the search procedure and are also accumulated in an accumulator array. The next BP is selected as that for which the (accumulated) distance is the largest. The procedure continues computing the distances from the successively selected BPs to the other prototypes, storing the computed distances, accumulating them into the accumulator array, and selecting the next BP as the one for which the accumulated distance to the other already selected BPs is maximum. The stop condition is reached when a prespecified number of BPs and the corresponding distances have been obtained. A formal description of this procedure is given as the *BP-Selection Algorithm* below.

Algorithm BP-Selection

```
Input: $P \subset E$; $m \in \mathbb{N}$; {finite set of prototypes and number of BPs}
Output: $B \subseteq P$, $|B| = m$; {set of $m$ Base Prototypes (BPs)}
  $D \in \mathbb{R}^{|P| \times |B|}$; {$|P| \cdot |B|$ interprototype distances}
Function: $d : E \times E \to \mathbb{R}$; {distance function}
Variables: $A \in \mathbb{R}^{|P|}$; {distance accumulator array}
  $b, b' \in P$; $max \in \mathbb{R}$;

begin
  $b'$ := arbitrary_element($P$); $B := \{b'\}$; $A := [0]$;
  while $|B| < m$ do
    $max := 0$; $b := b'$;
    for every $p \in P - B$ do
      $D[b, p] := d(b, p)$;
      $A[p] := A[p] + D[b, p]$;
      if ($A[p] > max$) then $b' := p$; $max := A[p]$; endif
    endfor
    $B := B \cup \{b'\}$;
  endwhile
end
```

The computational burden of this algorithm is clearly seen to be $n \cdot m$ steps (each involving one distance computation and other elementary unit-cost operations), where $n = |P|$ is the number of prototypes and $m = |B|$ is the given number of Base Prototypes. It should be noted, however, that this procedure does not guarantee a strict optimality (maximum separation of BPs). Nevertheless, as will be shown later, good results are obtained with BPs selected in this way, therefore making fairly unnecessary the (otherwise probably infeasible) search for a truly optimal set of BPs.

3. The searching algorithm

Once a matrix of distances from BPs to the remaining prototypes is available, the Linear AESA (LAESA) NN searching strategy is similar to that of the original AESA. More specifically, the LAESA can be derived as a Branch and Bound algorithm quite in the same way as AESA was in Vidal (1994). The main difference is that, now, the bounding function can no longer rely on the whole set of interprototype distances and it should rather be based on BPs distances alone.

Let P be the set of prototypes and $B \subseteq P$ the set of BPs. Let x be a test sample and $Q \subseteq P$ be a set of prototypes q for which $d(x, q)$ has already been computed (and stored) in previous steps of the search procedure. Then for every $p \in P$, the following lower bound estimation $g_Q(p)$ of the distance from x to p can be easily derived from the *triangle inequality* of $d(\bullet, \bullet)$ (Vidal (1992)):

$$d(x, p) \geqslant g_Q(p)$$

$$= \begin{cases} 0 & \text{if } Q \cap B = \emptyset, \\ \max\limits_{\forall q \in Q \cap B} |d(p, q) - d(x, q)| & \text{otherwise.} \end{cases} \tag{1}$$

This lower bound can be cheaply computed since both $d(p, q)$ $\forall p \in P$, $\forall q \in B$ and $d(x, q)$ $\forall q \in Q$ are readily available (in constant time). Therefore, we can directly and efficiently eliminate every prototype, p, for which the lower-bound (optimistic) estimation (1) of its not-yet-computed distance to x is worse than the best true distance computed so far. On the other hand, by adopting *the lower-bound driven best-first strategy of Branch & Bound* (which is generally recognized to often lead to the fastest procedures), *the same* bounding function can be also used as a convenient estimator of true distances to drive the AESA 'Approximating Step' (Vidal (1994)): i.e., select each next candidate prototype, p, as that for which $g_Q(p)$ is the smallest. This directly results in a basic version of LAESA. Nevertheless, some additional details arise here with regard to the *management of BPs*. In particular, since (unlike it happens in the approaches of Ramasubramanian (1991) and Paliwal) B is a subset of P, BPs should be candidates not only to selection, but also to elimination or pruning. Early selection of BPs can be seen as advantageous, since they can effectively help tightening the lower bounds of the remaining non-eliminated prototypes. Similarly, early elimination of BPs can be considered unfavorable. Nevertheless, for each configuration of P and for each test sample, the usefulness of different BPs can be diverse and one would also admit (early) eliminating BPs if they are not likely to help tightening the lower bounds.

Taking these considerations into account, the proposed algorithm (given below) makes use of two generic functions, CONDITION and CHOICE, to allow for different strategies of BP management. The algorithm repeatedly performs five basic steps until all prototypes have been eliminated: *Distance Computing, Updating the nearest-to-x prototype, Updating Lower Bounds, Approximating* and *Eliminating*. The first iteration starts with a *Distance Computation* from x to an arbitrarily selected BP and the corresponding *Updating of the Lower Bounds* of the remaining prototypes. This last step is executed only for BPs, since only for these prototypes the distances to the other prototypes are available in the precomputed matrix, D. On the basis of these Lower Bounds, the *Approximating* step successively selects non-eliminated prototypes from B as long as certain conditions are met; otherwise, a non-eliminated prototype from $P-B$ is selected. The CHOICE function helps deciding whether a base or non-base prototype is to be selected. The *Elimination* of PBs, on the other hand, is mainly controlled by the boolean function CONDITION. Some basic implementations of the CHOICE and CONDITION functions are given below and their effectiveness are compared through simulation experiments. It should be noted that both *Approximating* and *Eliminating*, as well as *Lower Bounds Updating* can be integrated into a single loop '*for every* $p \in P$'. Like in the last AESA version (Vidal (1994)), this leads to a more clear presentation of the resulting algorithm and also helps reducing the *overhead* (computation not alloted to distance computations).

The computational cost of LAESA directly depends on the number of iterations of the main *while* loop. Simulation experiments presented below clearly suggest that, for large data sets and fixed number of BPs, this number tends to be a small constant on the average (i.e., independent of the total number of prototypes). Correspondingly, the asymptotical *average number of distance computations* will also be *constant*, as it similarly happens in the previous AESA versions (Vidal (1986, 1994), Ramasubramanian (1991)). The overall computing time, on the other hand, also entails computation not alloted to distance computations. Such a computation or *overhead* is clearly $O(n \cdot m)$, where $n = |P|$ and $m = |B|$. Obviously, for large data sets and computationally cheap distances, this linear overhead may effectively outweigh the cost of the small (constant) number of distance computations themselves. While not very important in a large number of practical applications, this remaining bottleneck of all AESA-related strategies still prevents their use in very large problems and future work should obviously address this problem.

Finally, with respect to the generic functions CHOICE and CONDITION, only some basic policies have been explored in this work. In all of them, a single CHOICE function has been used which simply consists of selecting BPs whenever possible; that is:

$$\text{CHOICE}(b, q) = (\textit{if } b \neq \text{indeterminate } \textit{then } b \ \textit{else } q \ \textit{endif}).$$

On the other hand, a simple family of boolean functions CONDITION have been considered in which elimination of BPs is allowed only after having previously selected a number of (Base) prototypes, nc, greater than a specified fraction, k, of the original number of BPs, m; that is:

$$\text{CONDITION} = (nc > m/k) .$$

For values of $k=1, 2, ..., \infty$, this function leads to the BP management policies hereafter referred to as EC1, EC2, ..., EC∞.

While these simple strategies have been found to lead to satisfactory results (see Section 4), many other more sophisticated strategies are possible. Some of these possibilities are currently being explored (Micó et al. (1992)), and comprehensive results will be reported in future papers.

Algorithm LAESA

```
Input  $P \subset E$, $n = |P|$;  {finite set of prototypes}
   $B \subseteq P$, $m = |B|$;  {set of Base Prototypes}
   $D \in \mathbb{R}^{n\times m}$;  {precomputed $n \times m$ array of interprototype distances}
   $x \in E$;  {test sample}
Output  $p^* \in P$; $d^* \in \mathbb{R}$;  {nearest neighbour prototype and its distance to $x$}
Functions:  $d: E \times E \to \mathbb{R}$;  {distance function}
   CONDITION : Boolean;  {controls the elimination of Base Prototypes}
   CHOICE : $B \times (P-B) \to P$;  {selection of Base or non-Base Prototypes}
Variables:  $p, q, s, b \in P$;
   $G \in \mathbb{R}^n$;  {lower bounds array}
   $dxs, gp, gq, gb \in \mathbb{R}$;
   $nc \in \mathbb{N}$;  {number of computed distances}

begin
 $d^* := \infty$; $p^* :=$ indeterminate; $G := [0]$;
 $s :=$ arbitrary_element$(B)$; $nc := 0$;
 while $|P| > 0$ do
   $dxs := d(x, s)$; $P := P - \{s\}$; $nc := nc + 1$;
     {distance computing}
   if $dxs < d^*$ then $p^* := s$; $d^* := dxs$; endif
     {updating $p^*$, $d^*$}
   $q :=$ indeterminate;  $gq := \infty$;  $b :=$ indeterminate;
   $gb := \infty$;
   for every $p \in P$ do  {eliminating and approximating loop}
     if $s \in B$ then  {updating $G$, if possible}
       $G[p] := \max(G[p], |D[p, s] - dxs|)$
     endif
     $gp := G[p]$;
     if $p \in B$ then
       if ($gp \geq d^*$ & CONDITION) then $P := P - \{p\}$
         {eliminating from $B$}
       else  {approximating: selecting from $B$}
         if $gp < gb$ then $gb := gp$; $b := p$ endif
       endif
     else
       if $gp \geq d^*$ then $P := P - \{p\}$  {eliminating from $P-B$}
       else  {approximating: selecting from $P-B$}
         if $gp < gq$ then $gq := gp$; $q := p$ endif
       endif
     endif
   endfor
   $s :=$ CHOICE$(b, q)$;
 endwhile
end
```

4. Experiments

The performance of LAESA was determined through a series of simulation experiments in abstract metric spaces. In all the cases, both prototypes and test samples were drawn from uniform distributions in *D-dimensional hypercubes*. The coordinates of these data points where never used directly and they only served to measure interpoint distances, using an appropriate metric which, in this case, was the Euclidean *D*-dimensional norm. Results with other usual metrics were similar to those reported below and are omitted here for the sake of brevity.

The experimental procedure to obtain each forthcoming result was as follows: after having drawn a random set of *n prototypes*, the *BP-selection* algorithm was run to obtain a prespecified number, *m*, of *Base Prototypes*, where *m* was varied in most of the experiments. Then, an independent random set of *1000 test points* was drawn from the same uniform distribution and the *LAESA* was applied to each of these points. The results (typically number of distance computations required, *nc*) were averaged over the 1000 test points. Moreover, in order to obtain sound statistical results, the same procedure was always run 10 times with independent random prototype and test sets, and the results were averaged over these 10 trials. Therefore, each datum plotted in the figures below corresponds to an overall 10^4 LAESA Nearest-Neighbour searches, with 10 different random configurations of Prototypes and the corresponding 10 configurations of BPs as selected by the *BP-SELECTION* algorithm.

The first experiment was devoted to compare the relative performance of different Base-Prototype management policies. The experiment involved $n = 1024$ prototypes and the Euclidean metric in dimension $D = 6$. The number of Base Prototypes, m, was appropriately varied within the 5 to 100 range. The results, using the above indicated experimental procedure, are shown in Figure 1. It is worth noting that (excluding EC∞) an *optimal number of BPs*, is observed for all the policies (EC1, EC2, ...). Also, a range of adequate values of m around the optimal one can be used with nearly optimal performance. This range tends to broaden as increased freedom is granted to LAESA to eliminate its Base Prototypes (EC2, EC3, ...), with the broadest region corresponding to EC∞. In this case, Base Prototypes are not distinguished from ordinary prototypes for elimination purposes and, obviously, the optimal value tends to be obtained for $m = n$, with the same performance as with the conventional AESA. While having a broad range of adequate number of BPs is an obvious convenient feature, the performance of EC∞ is significantly worse than that of the other strategies if a small number of BPs is adopted. Therefore, one should better use ECk strategies with $k < \infty$, in order to trade computational performance for freedom in the choice of the number of BPs. In any case, from the overall results of Figure 1, it is clearly seen that a search efficiency of LAESA ($nc \approx 20$) close to that of AESA ($nc \approx 16$) can be easily achieved with a very small number of Base Prototypes ($m < 100 \ll 1024 = n$).

In the second experiment, only the EC1 BP-selection policy was used but different dimensions were considered. The number of prototypes was also set in this case to $n = 1024$ and, following the usual experimental procedure, the results of Figure 2 were obtained. These results indicate a similar behavior for all the dimensions, with a varying '*optimal range*' in the number of BPs. In particular, the greater the dimension, the greater is the optimal number of BPs; but also the less critical (i.e., the broadest) is the corresponding range.

The above experiment was further repeated several times, for varying number of prototypes, n, in order to establish to which extent the optimal number of BPs (m^*) depends on how large is the problem considered (n). For this purpose, after running each whole experiment for a given value of n and different

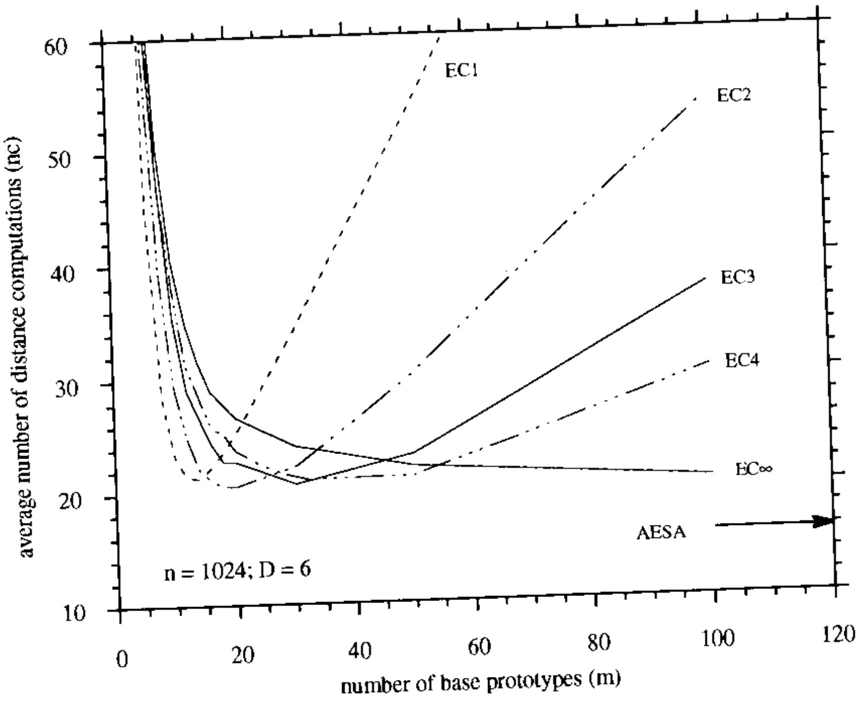

Figure 1. Average number of distance computations (nc) required by LAESA with different Base-Prototype management policies, as a function of the number of Base Prototypes (m), for a random set of $n = 1024$ 6-dimensional prototype vectors using the Euclidean Metric.

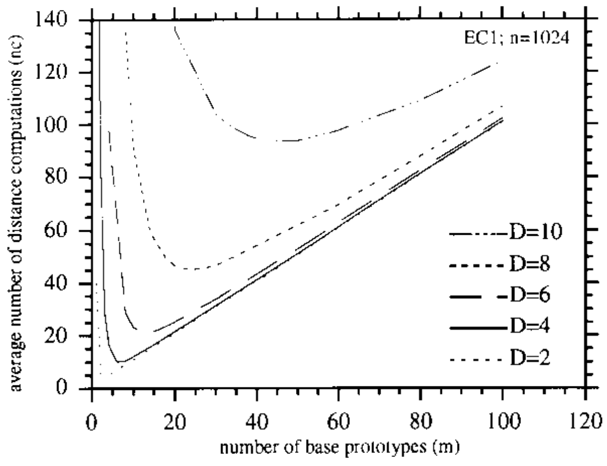

Figure 2. Average number of distance computations (nc) required by EC1-LAESA, as a function of the number of Base Prototypes (m) and different dimensions, using the Euclidean Metric and a number of prototypes $n = 1024$.

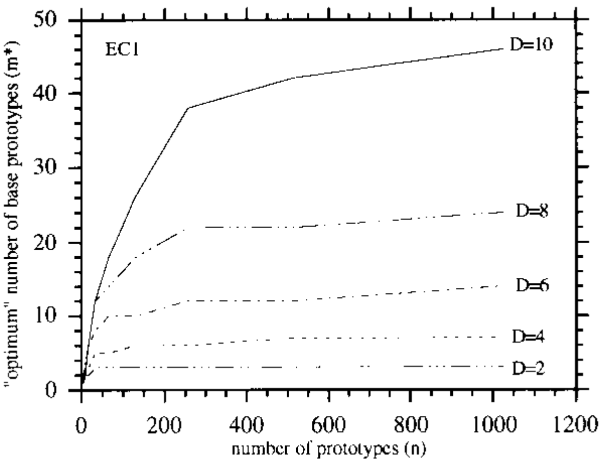

Figure 3. 'Optimal' number of Base Prototypes (m^*) for EC1-LAESA, as a function of the size of complete set of prototypes (n) and different dimensions using the Euclidean Metric.

dimensions, D, the corresponding m^* value was determined to obtain the results plotted in Figure 3. It is very interesting to note in this figure that, for large sets of prototypes, *no* significant variation is observed in the optimal number of Base Prototypes. From a practical point of view, this adds a considerable freedom in the choice of an adequate size for the set of BPs. But, even more important, this behavior, along with that shown in the next experiment, allows us to properly claim that, like AESA, the LAESA also works with *asymptotical average constant number of distance computations*; i.e., computation neither dependent on the number of prototypes nor on the number of Base Prototypes.

The next experiment was carried out in order to directly show the actual dependence of LAESA performance on the number of prototypes, n, for different dimensions, D. For this purpose, the number of base prototypes, m, was fixed to the corresponding optimal value, m^*, determined in the previous experiment for the largest prototype set ($n = 1024$) in every dimension. Then, using the EC1 BP management policy, the usual experimental procedure was carried out for these dimensions and varying numbers of prototypes, n. The results, displayed in Figure 4, clearly show the claimed asymptotical constant average number of distance computations.

For the sake of comparison, some of these results are also displayed in Figure 5 along with the corresponding results of the conventional AESA. It is clear from this figure that the new LAESA computes less than 1.5 times more distances than AESA while requiring a dramatically smaller preprocessing time and memory space.

The results of Figure 4 may suggest that efficiency can exponentially decrease as dimension increases.

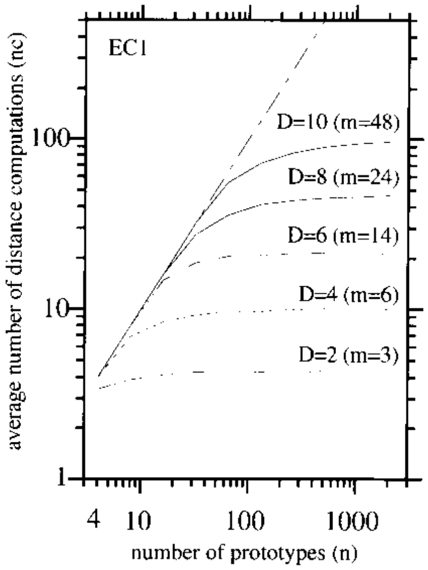

Figure 4. Average number of distance computations (nc) required by EC1-LAESA, as a function of the number of prototypes (n) and different dimensions, using the Euclidean Metric.

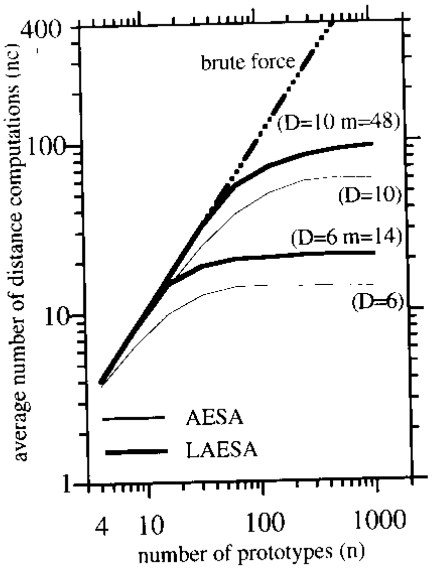

Figure 5. Comparison of some of the LAESA results of Figure 4 with the corresponding results of the original AESA. LAESA can consistently find Nearest Neighbours by computing only less than 1.5 more average distances than AESA, while requiring orders of magnitude less memory space and preprocessing time.

However, one should take into account that, for a fixed number of uniformly distributed points in *D*-dimensional hypercubes, their average distances (rapidly) grow with dimension. Correspondingly, given the large tendency of AESA (Vidal (1986)) (and LAESA alike) to require greater number of distance computations as distances from test samples to their NN prototypes increase, the apparent efficiency loss is actually due to the rapid increase of average distances as the unit hypercube exponentially decreases its volume. Like in the conventional AESA (Vidal (1986)), this behavior is not expected for real, clustered data, where average distances from test samples to their corresponding prototypes are not significantly increased as more features are added to the representation of objects. Also, one should take into account that (L)AESA directly works with *metric spaces*, where only the concept of *intrinsic dimensionality* (Pettis et al. (1979), Fukunaga (1990), Baydal et al. (1989)) would properly make sense and, for adequate distances, the intrinsic dimensionality does not necessarily increase as larger (and better) representations of the objects are used.

5. Conclusions

The results reported in the last section clearly show that the new Linear Approximating and Eliminating Search Algorithm (LAESA) can be used for finding Nearest Neighbours in Metric Spaces with an average constant number of distance computations. This is achieved using a number of precomputed distances which only grows linearly with the number of prototypes. As compared with the quadratic memory requirements of the conventional AESA (Vidal (1986, 1994)), this represents a dramatic reduction of memory space and corresponding preprocessing time. From the results reported in this paper, such a memory cost reduction is achieved at the expense of a small increase in the number of distance computations with respect to those required by AESA searching. This corresponds to a particular family of (simple) Base Prototype management policies, but many other policies suggest themselves. For instance, BPs can be allowed to be eliminated if they are found not to help improving lower bounds in previous maps (Micó et al. (1992)). Also, the selection (choice) of a *base* or *non-base* prototype can be conditioned to the actual approximate distances (lower bounds) from the prototypes to the test sample. Using (some combination of) these policies, the (small) gap between the number of computation required by LAESA and AESA searching could perhaps be reduced. Finally, further improvements could be achieved by taking into account the ideas introduced in Shasha and Wang (1990) to take the maximum advantage of all the distances available.

Apart from these studies, a main issue that deserves further investigation concerns the *overhead*, or computation that is not strictly applied to distance computation. While the (L)AESA linear overhead seems not very important for a number of real-world applications involving complex (and computationally expensive) distances and moderately large sets of prototypes, it can in fact become a serious bottleneck if the number of prototypes becomes very large; say, over the millions. In order to reduce the overhead, the pruning or elimination of prototypes must be organized in such a way that a (large) number of prototypes can be eliminated in a single step. One possibility for this could arise from the so called '*Spherical Grid*' AESA formulation of Ramasubramanian (1991) and Ramasubramanian and Paliwal (1992). But perhaps the most direct approach would result by organizing the set of prototypes into a tree structure; like, e.g., that used in Fukunaga and Narendra (1975). The goal now should be to combine the high approximating and eliminating efficiency of the (L)AESA with the low, typically logarithmic overhead that is possible with a tree organization of the prototypes.

References

Baydal, E., G. Andreu and E. Vidal (1989). Estimating the intrinsic dimensionality of discrete utterances. *IEEE Trans. Acoust. Speech Signal Process.* 37(5), 755–757.

Burkhard, W.A. and R.M. Keller (1973). Some approaches to best match file searching. *Comm. Ass. Comput. Mach.* 16, 230–236.

Casacuberta, F. and E. Vidal (1987). *Reconocimiento Automático del Habla.* Marcombo.

Dasarathy, B. (1991). *Nearest Neighbour (NN) norms: NN Pattern Classification Techniques.* IEEE Computer Soc. Press, Silver Spring, MD.

Fukunaga, K. and M. Narendra (1975). A branch and bound algorithm for computing *K*-nearest neighbours. *IEEE Trans. Comput.* 24, 750–753.

Fukunaga, K. (1990) *Introduction to Statistical Pattern Recognition.* Academic Press, New York.

Marzal, A. and E. Vidal (1992). Computation of normalized edit distance and applications. *IEEE Trans. Pattern Anal. Machine Intell.*, to appear.

Micó, M.L., J. Oncina and E. Vidal (1991). Algoritmo para encontrar el vecino más próximo en un tiempo medio constante con una complejidad espacial lineal. Tech. Report DSIC II/14-91, Universidad Politécnica de Valencia.

Micó, M.L., J. Oncina and E. Vidal (1992). An algorithm for finding nearest neighbours in constant average time with a linear space complexity. *Proc. 11th ICPR*, The Hague, Vol. II, 557–560.

Pettis, K.W., T.A. Bailey, A.K. Jain and R.C. Dubes (1979). An intrinsic dimensionality estimator from near-neighbour information. *IEEE Trans. Pattern Anal. Machine Intell.* 1, 41.

Rabiner, L. and S.E. Levinson (1984). Isolated and connected word recognition — Theory and selected applications. *IEEE Trans. Comm.* 29, 621–659.

Ramasubramanian, V. and K.K. Paliwal (1990). An efficient approximation-elimination algorithm for fast nearest-neighbour search based on a spherical distance coordinate formulation. In: L. Torres et al., Eds., *Signal Processing V: Theories and Application.* Proc. EUSIPCO-90. North-Holland, Amsterdam.

Ramasubramanian, V. (1991). Fast Algorithms for Nearest-Neighbour Search and Application to Vector Quantization, PhD dissertation. University of Bombay.

Ramasubramanian, V. and K.K. Paliwal (1992). An efficient approximation-elimination for fast nearest-neighbor search. *ICASSP*-92.

Sanfeliu, A. and K.S. Fu (1983). A distance measure between attributed graphs for pattern recognition. *IEEE Trans. Syst. Man Cybernet.* 13, 353–362.

Shapiro, M. (1977). The choice of reference points in best-match file searching. *Artificial Intelligence/Language Process.* 20, 339–343.

Shapiro, L.G. and R.M. Haralik (1985). A metric for comparing relational descriptions. *IEEE Trans. Pattern Anal. Machine Intell.* 7, 90–94.

Shasha, D. and T. Wang (1990): New techniques for best-match retrieval. *ACM Trans. Inform. Syst.* 8(2), 140–158.

Vidal, E. (1986). An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recognition Lett.* 4(3), 145–157.

Vidal, E., H. Rulot, F. Casacuberta and J.M. Benedí (1988). On the use of a metric-space search algorithm — AESA — for fast DTW-based recognition of isolated words. *IEEE Trans. Acoust. Speech Signal Process.* 36(5), 651–660.

Vidal, E. and M.J. Lloret (1988). Fast speaker independent DTW recognition of isolated words using a metric space search algorithm (AESA). *Speech Communication* 7, 417–422.

Vidal, E. (1994). New formulation and improvements of the Nearest-Neighbour Approximating and Eliminating Search Algorithm (AESA). *Pattern Recognition Lett.* 15, 1–7.