On The Marriage of Lp-norms and Edit Distance

Lei Chen
School of Computer Science
University of Waterloo
l6chen@uwaterloo.ca

Raymond Ng
Department of Computer Science
University of British Columbia
rng@cs.ubc.ca

Abstract

Existing studies on time series are based on two categories of distance functions. The first category consists of the Lp-norms. They are metric distance functions but cannot support local time shifting. The second category consists of distance functions which are capable of handling local time shifting but are non-metric. The first contribution of this paper is the proposal of a new distance function, which we call ERP ("Edit distance with Real Penalty"). Representing a marriage of L1-norm and the edit distance, ERP can support local time shifting, and is a metric.

The second contribution of the paper is the development of pruning strategies for large time series databases. Given that ERP is a metric, one way to prune is to apply the triangle inequality. Another way to prune is to develop a lower bound on the ERP distance. We propose such a lower bound, which has the nice computational property that it can be efficiently indexed with a standard B+-tree. Moreover, we show that these two ways of pruning can be used simultaneously for ERP distances. Specifically, the false positives obtained from the B+-tree can be further minimized by applying the triangle inequality. Based on extensive experimentation with existing benchmarks and techniques, we show that this combination delivers superb pruning power and search time performance, and dominates all existing strategies.

Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

1 Introduction

Many applications require the retrieval of similar time series. Examples include financial data analysis and market prediction [1, 2, 10], moving object trajectory determination [6] and music retrieval [31]. Studies in this area revolve around two key issues: the choice of a distance function (similarity model), and the mechanism to improve retrieval efficiency.

Concerning the first issue, many distance functions have been considered, including Lp-norms [1, 10], dynamic time wraping (DTW) [30, 18, 14], longest common subsequence (LCSS) [4, 25] and edit distance on real sequence (EDR) [6]. Lp-norms are easy to compute. However, they cannot handle local time shifting, which is essential for time series similarity matching. DTW, LCSS and EDR have been proposed to exactly deal with local time shifting. However, they are non-metric distance functions.

This leads to the second issue of improving retrieval efficiency. Specifically, non-metric distance functions complicate matters, as the violation of the triangle inequality renders most indexing structures inapplicable. To this end, studies on this topic propose various lower bounds on the actual distance to guarantee no false dismissals [30, 18, 14, 31]. However, those lower bounds can admit a high percentage of false positives.

In this paper, we consider both issues and explore the following questions.

- *Is there a way to combine Lp-norms and the other distance functions so that we can get the best of both worlds* – namely being able to support local time shifting and being a metric distance function?
- With such a metric distance function, we can apply the triangle inequality for pruning, but *can we develop a lower bound for the distance function?* If so, is lower bounding more efficient than applying the triangle inequality? Or, is it possible to do both?

Our contributions are as follows:

- We propose in Section 3 a distance function which we call *Edit distance with Real Penalty* (ERP). It

can be viewed as a variant of L1-norm, except that it can support local time shifting. It can also be viewed as a variant of EDR and DTW, except that it is a metric distance function. We present benchmark results showing that this distance function is natural for time series data.

- We propose in Section 4 a new lower bound for ERP, which can be efficiently indexed with a standard B+-tree. Given that ERP is a metric distance function, we can also apply the triangle inequality. We present benchmark results in Section 5 comparing the efficiency of lower bounding versus applying the triangle inequality.
- Last but not least, we develop in Section 4 a k-nearest neighbor (k-NN) algorithm that applies both lowering bounding and the triangle inequality. We give extensive experimental results in Section 5 showing that this algorithm gets the best of both paradigms, delivers superb retrieval efficiency and dominates all existing strategies.

2 Related Work

Many studies on similarity-based retrieval of time series were conducted in the past decade. The pioneering work by Agrawal et al. [1] used Euclidean distance to measure similarity. Discrete Fourier Transform (DFT) was used as a dimensionality reduction technique for time series data, and an R-tree was used as the index structure. Faloutsos et al. [10] extended this work to allow subsequence matching and proposed the GEMINI framework for indexing time series. The key is the use of a lower bound on the true distance to guarantee no false dismissals when the index is used as a filter.

Subsequent work have focused on two main aspects: new dimensionality reduction techniques (assuming that the Euclidean distance is the similarity measure); and new approaches for measuring the similarity between two time series. Examples of dimensionality reduction techniques include Single Value Decomposition [19], Discrete Wavelet Transform [20, 22], Piecewise Aggregate Approximation [15, 29], and Adaptive Piecewise Constant Approximation [14].

The motivation for seeking new similarity measures is that the Euclidean distance is very weak on handling noise and local time shifting. Berndt and Clifford [3] introduced DTW to allow a time series to be "stretched" to provide a better match with another time series. Das et al. [9] and Vlachos et al. [25] applied the LCSS measure to time series matching. Chen et al. [6] applied EDR to trajectory data retrieval and proposed a dimensionality reduction technique via a symbolic representation of trajectories. However, none of DTW, LCSS and EDR is a metric distance function for time series.

Most of the approaches on indexing time series follow the GEMINI framework. However, if the distance measure is a metric, then existing indexing structures proposed for metrics may be applicable. Examples include the MVP-tree [5], the M-tree [8], the Sa-tree [21], and the OMNI-family of access methods [11]. A survey of metric space indexing is given in [7]. In our experiments, we pick M-trees and OMNI-sequential as the strawman structures for comparison; MVP-trees and Sa-trees are not compared because they are main memory resident structures. The other access methods of OMNI-family are not used because the dimensionality of OMNI-coordinates is high (e.g., ≥ 20), which may lead to dimensionality curse [28]. In general, a common strategy to apply the triangle inequality for pruning is to use a set of reference points (time series in this case). Different studies propose different ways to choose the reference points. In our experiments, we compare our strategies in selecting reference points with the HF algorithm of the OMNI-family.

Symbols	Meaning
S	a time series $[s_1, \ldots, s_n]$
$Rest(S)$	$[s_2, \ldots, s_n]$
$dist(s_i, r_i)$	the distance between two elements
$\hat{S}$	S after aligned with another series
DLB	a lower bound of the distance

Figure 1: Meanings of Symbols Used

3 Edit Distance With Real Penalty

3.1 Reviewing Existing Distance Functions

A *time series* S is defined as a sequence of real values, with each value s_i sampled at a specific time, i.e., $S = [s_1, s_2, \ldots, s_n]$. The *length* of S is n, and the n values are referred to as the n *elements*. This sequence is called the *raw representation* of the time series. Given S, we can normalize it using its mean (μ) and standard deviation (σ) [13]: $Norm(S) = [\frac{s_1-\mu}{\sigma}, \frac{s_2-\mu}{\sigma}, \ldots, \frac{s_n-\mu}{\sigma}]$. Normalization is recommended so that the distance between two time series is invariant to amplitude scaling and (global) shifting of the time series. Throughout this paper, we use S to denote $Norm(S)$ for simplicity, even though all the results developed below apply to the raw representation as well. Figure 1 summarizes the main symbols used in this paper.

Given two time series R and S of the same length n, the L1-norm distance between R and S is: $\sum_{i=1}^{n} dist(r_i, s_i) = \sum_{i=1}^{n} |r_i - s_i|$. This distance function satisfies the triangle inequality and is a metric. The problem in using L1-norm for time series is that it requires the time series to be of the same length and does not support local time shifting.

To cope with local time shifting, one can borrow ideas from the domain of strings. A string is a sequence of elements, each of which is a symbol in an alphabet. Two strings, possibly of different lengths, are aligned so that they become identical with the smallest number of added, deleted or changed symbols. Among these three operations, deletion can be

treated as adding a symbol in the other string. Hereafter, we refer to an added symbol as a *gap* element. This distance is called the *string edit distance*. The cost/distance of introducing a gap element is set to 1.

$$dist(r_i, s_i) = \begin{cases} 0 & \text{if } r_i = s_i \\ 1 & \text{if } r_i \text{ or } s_i \text{ is a gap} \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

In the above formula, we highlight the second case to indicate that if a gap is introduced in the alignment, the cost is 1. String edit distance satisfies the triangle inequality and is a metric [27].

To generalize from strings to time series, the complication is that the elements r_i and s_i are not symbols, but real values. For most applications, strict equality would not make sense as, for instance, the pair $r_i = 1, s_i = 2$ should be considered more similar than the pair $r_i = 1, s_i = 10000$. To take the real values into account, one way is to relax equality to be within a certain tolerance δ:

$$dist_{edr}(r_i, s_i) = \begin{cases} 0 & \text{if } |r_i - s_i| \leq \delta \\ 1 & \text{if } r_i \text{ or } s_i \text{ is a gap} \\ 1 & \text{otherwise} \end{cases} \quad (2)$$

This is a simple generalization of Formula (1). Based on Formula (2) on individual elements and gaps, the *edit distance* between two time sequences R and S of length m and n respectively is defined in [6] as Formula (3) in Figure 2. r_1 and $Rest(R)$ denote the first element and the remaining sequence of R respectively. Notice that given Formula (2), the last case in Formula (3) can be simplified to: $min\{EDR(Rest(R), Rest(S)) + 1, EDR(Rest(R), S)+1, EDR(R, Rest(S))+1\}$. Local time shifting is essentially implemented by a dynamic-programming style minimization of the above three possibilities.

While EDR can handle local time shifting, it no longer satisfies the triangle inequality. The problem arises precisely from relaxing equality, i.e., $|r_i - s_i| \leq \delta$. More specifically, for three elements q_i, r_i, s_i, we can have $|q_i - r_i| \leq \delta$, $|r_i - s_i| \leq \delta$, but $|q_i - s_i| > \delta$.

To illustrate, let us consider a very simple example of three time series: $Q = [0], R = [1, 2]$ and $S = [2, 3, 3]$. Let $\delta = 1$. To best match R, Q is aligned to be $\widetilde{Q} = [0, -]$, where the symbol "-" denotes a gap. (There may exist many alternative ways to align sequences to get their best match. We only show one of the possible alignments for simplicity.) Thus, $EDR(Q, R) = 0 + 1 = 1$. Similarly, to best match S, R is aligned to be $\widetilde{R} = [1, 2, -]$, giving rise to $EDR(R, S) = 1$. Finally, to best match S, Q is aligned to be $\widetilde{Q} = [0, -, -]$, leading to $EDR(Q, S) = 3 > EDR(Q, R) + EDR(R, S) = 1 + 1 = 2$!

DTW differs from EDR in two key ways, summarized in the following formula:

$$dist_{dtw}(r_i, s_i) = \begin{cases} |r_i - s_i| & \text{if } r_i, s_i \text{ not gaps} \\ |r_i - s_{i-1}| & \text{if } s_i \text{ is a gap} \\ |s_i - r_{i-1}| & \text{if } r_i \text{ is a gap} \end{cases} \quad (6)$$

First, unlike EDR, DTW does not use a δ threshold to relax equality, the actual L1-norm is used. Second, unlike EDR, there is no explicit gap concept being introduced in its original definition [3]. We treat the replicated elements during the process of aligning two sequences as gaps of DTW. Therefore, the cost of a gap is not set to 1 as EDR does; it amounts to replicating the previous element, based on which the L1-norm is computed. Based on the above formula, the dynamic warping distance between two time series, denoted as $DTW(R, S)$, is defined formally as Formula (4) in Figure 2. The last case in the formula deals with the possibilities of replicating either s_{i-1} or r_{i-1}.

Let us repeat the previous example with DTW: $Q = [0], R = [1, 2]$ and $S = [2, 3, 3]$. To best match R, Q is aligned to be $\widetilde{Q} = [0, -] = [0, 0]$. Thus, $DTW(Q, R) = 1 + 2 = 3$. Similarly, to best match S, R is aligned to be $\widetilde{R} = [1, 2, -] = [1, 2, 2]$, giving rise to $DTW(R, S) = 3$. Finally, to best match S, Q is aligned to be $\widetilde{Q} = [0, -, -] = [0, 0, 0]$, leading to $DTW(Q, S) = 8 > DTW(Q, R) + DTW(R, S) = 3 + 3 = 6$.

It has been shown in [24] that for speech applications, DTW "loosely" satisfies the triangle inequality. We verified this observation with the 24 benchmark data sets used in [14, 31]. It appears that this observation is not true in general, as on average nearly 30% of all the triplets do not satisfy the triangle inequality.

3.2 ERP and its Properties

The key reason why DTW does not satisfy the triangle inequality is that, when a gap needs to be added, it replicates the previous element. Thus, as shown in the second and third cases of Formula (6), the difference between an element and a gap varies according to r_{i-1} or s_{i-1}. Contrast this situation with EDR, which makes every difference to be a constant 1 (second case in Formula (2)). On the other hand, the problem for EDR lies in its use of a δ tolerance. DTW does not have this problem because it uses the L1-norm between two non-gap elements.

We propose ERP such that it uses real penalty between two non-gap elements, but a constant value for computing the distance for gaps. Thus, ERP uses the following distance formula:

$$dist_{erp}(r_i, s_i) = \begin{cases} |r_i - s_i| & \text{if } r_i, s_i \text{ not gaps} \\ |r_i - g| & \text{if } s_i \text{ is a gap} \\ |s_i - g| & \text{if } r_i \text{ is a gap} \end{cases} \quad (7)$$

where g is a constant value. Based on Formula (7), we define the ERP distance between two time series,

$$EDR(R,S) = \begin{cases} n & \text{if } m = 0 \\ m & \text{if } n = 0 \\ EDR(Rest(R), Rest(S)) & \text{if } dist_{edr}(r_1, s_1) = 0 \\ min\{EDR(Rest(R), Rest(S)) + dist_{edr}(r_1, s_1), & \text{otherwise} \\ EDR(Rest(R), S) + dist_{edr}(r_1, gap), EDR(R, Rest(S)) + dist_{edr}(gap, s_1)\} & \end{cases} \quad (3)$$

$$DTW(R,S) = \begin{cases} 0 & \text{if } m = n = 0 \\ \infty & \text{if } m = 0 \text{ or } n = 0 \\ dist_{dtw}(r_1, s_1) + min\{DTW(Rest(R), Rest(S)), & \text{otherwise} \\ DTW(Rest(R), S), DTW(R, Rest(S))\} & \end{cases} \quad (4)$$

$$ERP(R,S) = \begin{cases} \sum_1^n |s_i - g| & \text{if } m = 0 \\ \sum_1^m |r_i - g| & \text{if } n = 0 \\ min\{ERP(Rest(R), Rest(S)) + dist_{erp}(r_1, s_1), & \text{otherwise} \\ ERP(Rest(R), S) + dist_{erp}(r_1, gap), ERP(R, Rest(S)) + dist_{erp}(s_1, gap)\} & \end{cases} \quad (5)$$

Figure 2: Comparing the Distance Functions

denoted as $ERP(R,S)$, as Formula (5) in Figure 2. A careful comparison of the formulas reveals that ERP can be seen as a combination of L1-norm and EDR. ERP differs from EDR in avoiding the δ tolerance. On the other hand, ERP differs from DTW in not replicating the previous elements. The following lemma shows that for any fixed constant g, the triangle inequality is satisfied.

Lemma 1 For any three elements q_i, r_i, s_i, any of which can be a gap element, it is necessary that $dist(q_i, s_i) \leq dist(q_i, r_i) + dist(r_i, s_i)$ based on Formula (7).

Theorem 1 Let Q, R, S be three time series of arbitrary length. Then it is necessary that $ERP(Q,S) \leq ERP(Q,R) + ERP(R,S)$.

The proof of this theorem is a consequence of Lemma 1 and the proof of the result by Waterman et al. [27] on string edit distance. The Waterman proof essentially shows that defining the distance between two strings based on their best alignment in a dynamic programming style preserves the triangle inequality, as long as the underlying distance function also satisfies the triangle inequality. The latter requirement is guaranteed by Lemma 1. Due to lack of space, we omit a detailed proof.

3.2.1 Picking a Value for g

A natural question to ask here is: *what is an appropriate value of g?* The above lemma says that any value of g, as long as it is fixed, satisfies the triangle inequality. We pick $g = 0$ for two reasons. First, $g = 0$ admits an intuitive geometric interpretation. Consider plotting the time series with the x-axis representing (equally-spaced) time points and the y-axis representing the values of the elements. In this case, the x-axis corresponds to $g = 0$. Thus, the distance between two time series R, S corresponds to the difference between the area under R and the area under S.

Second, to best match R, S is aligned to form $\widetilde{S}$ with the addition of gap elements. However, since the gap elements are of value $g = 0$, it is easy to see that $\sum \widetilde{s}_i = \sum s_j$, making the area under S and that under $\widetilde{S}$ the same. The following lemma states this property. In the next section, we will see the computational significance of this lemma.

Lemma 2 Let R, S be two time series. By setting $g = 0$ in Formula (7), $\sum \widetilde{s}_i = \sum s_j$, where S is aligned to form $\widetilde{S}$ to match R.

Let us repeat the previous example with ERP: $Q = [0], R = [1,2]$ and $S = [2,3,3]$. To best match R, Q is aligned to be $\widetilde{Q} = [0,0]$. Thus, $ERP(Q,R) = 1+2 = 3$. Similarly, to best match S, R is aligned to be $\widetilde{R} = [1,2,0]$, giving rise to $ERP(R,S) = 5$. Finally, to best match S, Q is aligned to be $\widetilde{Q} = [0,0,0]$, leading to $ERP(Q,S) = 8 \leq ERP(Q,R) + ERP(R,S) = 3 + 5 = 8$, satisfying the triangle inequality.

To see how local time shifting works for ERP, let us change $Q = [3]$ instead. Then $ERP(Q,R) = 1+1 = 2$, as $\widetilde{Q} = [0,3]$. Similarly, $ERP(Q,S) = 2 + 3 = 5$, as $\widetilde{Q} = [0,3,0]$. The triangle inequality is satisfied as expected.

Notice that none of the results in this section are restricted to L1-norm. That is, if we use another Lp-norm to replace L1-norm in Formula (7), the lemma and the theorem remain valid. For the rest of the paper, we continue with L1-norm for simplicity.

3.3 On the Naturalness of ERP

Even though ERP is a metric distance function, it is a valid question to ask whether ERP is "natural" for time series. In general, whether a distance function is natural mainly depends on the application semantics. Nonetheless, we show two experiments below suggesting that ERP appears to be at least as natural as the existing distance functions.

The first experiment is a simple sanity check. We first generated a simple time series Q shown in Figure 3. Then we generated 5 other time series (T_1-T_5) by adding time shifting or noise data on one or two positions of Q as shown in Figure 3. For example, T_1 was generated by shifting the sequence values of Q to the left starting from position 4, and T_2 was derived from Q by introducing noise in position 4. Finally, we used L1-norm, DTW, EDR, ERP and LCSS to rank the five time series relative to Q. The rankings are listed left to right, with the leftmost being the most similar to Q. The rankings are as follow:

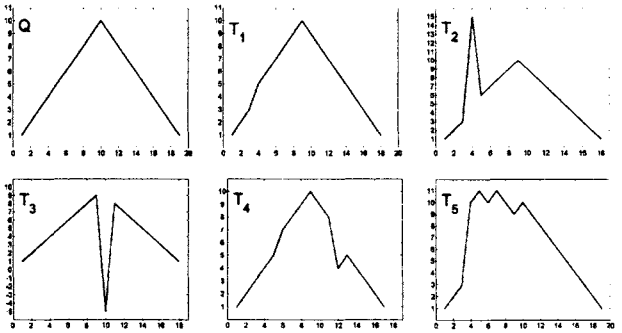


Figure 3: Subjective Evaluation of Distance Functions

L1-norm: T_1, T_4, T_5, T_3, T_2
LCSS: $T_1, \{T_2, T_3, T_4\}, T_5$
EDR: $T_1, \{T_2, T_3\}, T_4, T_5$
DTW: T_1, T_4, T_3, T_5, T_2
ERP: T_1, T_2, T_4, T_5, T_3

As shown from the above results, L1-norm is sensitive to noise, as T_2 is considered the worst match. LCSS focuses only on the matched parts and ignores all the unmatched portions. As such, it gives T_2, T_3, T_4 the same rank, and considers T_5 the worst match. EDR gives T_2, T_3 the same rank, higher than T_4. DTW gives T_3 a higher rank than T_5. Finally, ERP gives a ranked list different from all the others. Notice that the point here is *not* that ERP is the most natural. Rather, the point is that ERP appears to be no worse, if not better, than the existing distance functions.

In the second experiment, we turn to a more objective evaluation. Recently, Keogh et al. [17] have proposed using classification on labelled data to evaluate the efficacy of a distance function on time series. Specifically, each time series is assigned a class label. Then the "leave one out" prediction mechanism is applied to each time series in turn. That is, the class label of the chosen time series is predicted to be the class label of its nearest neighbour, defined based on the given distance function. If the prediction is correct, then it is a hit; otherwise, it is a miss. The classification error rate is defined as the ratio of the number of misses to the total number of the time series. In the table below, we show the average classification error rate using three benchmarks: the Cylinder-Bell-Funnel (CBFtr) data [12, 14], the ASL data [25] and the "cameramouse" (CM) data [25]. (All can be downloaded from *http://db.uwaterloo.ca/~l6chen/testdata*). Compared to the standard CBF data [17], temporal shifting is introduced in the CBFtr data set. The CBFtr data set is a 3-class problem. The ASL data set from UCI KDD archive consists of of signs from the Australian Sign Language. The ASL data set is a 10-class problem; The "cameramouse" data set contains 15 trajectories of 5 classes (words) (3 for each word). As shown in the table below, for three data sets, ERP performs (one of) the best, showing that it is not dominated by other well known alternatives.

Avg. Error Rate	L1	DTW	LCSS	EDR	ERP
CBFtr	0.03	0.01	0.01	0.01	0.01
ASL	0.16	0.10	0.11	0.11	0.09
CM	0.4	0.00	0.06	0.00	0.00

4 Indexing for ERP

Recall from Figure 2 that ERP can be seen as a variant of EDR and DTW. In particular, they share the same computational behavior. Thus, like EDR and DTW, it takes $O(mn)$ time to compute $ERP(Q, S)$ for time series Q, S of length m, n respectively. For large time series databases, it is important that for a given query Q, we try to minimize the computation of the true distance between Q and S for all series S in the database. The topic explored here is indexing for k-NN queries. An extension to range queries is rather straightforward; we omit details for brevity.

Given that ERP is a metric distance function, one obvious way to prune is to apply the triangle inequality. In Section 4.1, we present an algorithm to do just that. Metric or not, another common way to prune is to apply the GEMINI framework – that is, using lower bounds to guarantee no false negatives. Specifically, even though DTW is not a metric, three lower bounds have been proposed [30, 18, 14]. In Section 4.2.1, we show how to adapt these lower bounds for ERP. In Section 4.2.2, we propose a new lower bound for ERP. The beauty of this lower bound is that it can be indexed by a simple B+-tree.

4.1 Pruning by the Triangle Inequality

The procedure TrianglePruning shown in Figure 4 shows a skeleton of how the Triangle inequality is applied. The array *procArray* stores the true ERP distances computed so far. That is, if $\{R_1, \ldots, R_u\}$ is the set of time series for which $ERP(Q, R_i)$ has been computed, the distance $ERP(Q, R_i)$ is recorded in *procArray*. Thus, for time series S currently being evaluated, the triangle inequality ensures that $ERP(Q, S) \geq ERP(Q, R_i) - ERP(R_i, S)$, for all $1 \leq i \leq u$. Thus, it is necessary that $ERP(Q, S) \geq (max_{1 \leq i \leq u}\{ERP(Q, R_i) - ERP(R_i, S)\})$. This is implemented in lines 2 to 4. If this distance *maxPruneDist* is already worse than the current k-NN distance stored in *result*, then S can be skipped entirely. Otherwise, the true distance $ERP(Q, S)$ is computed, and *procArray* is updated to include S. Finally, the *result* array is updated, if necessary, to reflect the current k-NN neighbours and distances in sorted order.

The algorithm given in Figure 5 shows how the *result* and *procArray* should be initialized when the procedure TrianglePruning is called repeatedly in line 4. Line 3 of the algorithm represents a simple sequential scan of all the time series in the database. Note that we are *not* saying that a sequential scan should be used. We include it for two reasons. The first reason is to show how the procedure TrianglePruning can be

```
Procedure TrianglePruning(S, procArray, pmatrix, result, Q, k) {
  /* S ≡ the current time series; procArray ≡ the array of
  time series with computed true distance to Q; pmatrix ≡
  precomputed pairwise distance matrix; result ≡ the k-NN
  time series */
  (1) maxPruneDist = 0
  (2) for each time series R in procArray {
  (3)     if ( (procArray[R].dist − pmatrix[R, S]) > maxPruneDist)
  (4)         maxPruneDist = procArray[R].dist − pmatrix[R, S]
      } /* end-for, line 2 */
  (5) bestSoFar = result[k].dist /* the k-NN distance so far */
  (6) if (maxPruneDist ≤ bestSoFar) { /* cannot be pruned */
  (7)     realDist = ERP(Q, S) /* compute true distance */
  (8)     insert S and realDist into procArray
  (9)     if (realDist < bestSoFar) /* update result */
  (10)        insert S and realDist into result,
                sorted in ascending order of ERP distance
          } /* end-if, line 6 */
}
```

Figure 4: Algorithm for Applying the Triangle Inequality

```
Procedure SequentialScan(Q, k, pmatrix) {
  /* Q ≡ the query time series; pmatrix ≡
  precomputed pairwise distance matrix */
  (1) initialize result so that result[k].dist = ∞
  (2) initialize procArray to empty
  (3) for each time series S in the database
  (4)     TrianglePruning(S, procArray, pmatrix, result, Q, k)
  (5) return result
}
```

Figure 5: Sequential Scan Applying the Triangle Inequality

invoked. The second reason is that a sequential scan algorithm can be used to illustrate the *pure* pruning power of the triangle inequality, independent of the indexing structure. This issue will be discussed later when the inequality is used in conjunction with index structures. Moreover, the triangle inequality need not be applied only in the manner shown in Figure 4. It can be applied directly with existing metric space index structures such as M-trees or OMNI-family access methods. Again the issue of comparison will be addressed in Section 5.

Finally, for large databases, the procedure TrianglePruning makes two assumptions. The first assumption is that the matrix *pmatrix* is small enough to be contained in main memory. For every pair of time series R, S in the database, $pmatrix[R, S]$ records the distance $ERP(R, S)$. For large databases, *pmatrix* may be too large. The second assumption is that the size of *procArray* is small enough. The size is left unspecified because procedure TrianglePruning works regardless of the actual size of the array. The larger the size, the more time series can be used for pruning. These two assumptions will be addressed in Section 4.3.

4.2 Pruning by Lower Bounding

For non-metric distance functions, like DTW, the main pruning strategy is lower bounding. However, lower bounding is not restricted only to non-metric distance functions. Below we develop lower bounds for ERP.

4.2.1 Adapting Existing Lower Bounds

Given the similarities between ERP and DTW, as shown in Figure 2, we first adapt existing lower bounds for DTW to become lower bounds for ERP. Specifically, we consider the lower bounds proposed by Yi et al. [30], Kim et al. [18], and Keogh et al. [14].

The lower bound proposed by Yi et al., denoted as $DLB_{yi}(Q, S)$, is based on the area covered by Q that is greater than $max(S)$ and the area of Q that is less than $min(S)$, where $max(S)$ and $min(S)$ denote the maximum and minimum elements of S. Adapting to ERP is rather straightforward. Because ERP uses a constant gap element $g = 0$, we need to compare the original minimum and maximum elements with 0, i.e., $newmin(S) = min\{min(S), 0\}$ and $newmax(S) = max\{max(S), 0\}$. With this modification, we have the following result. The proof is omitted because it is a simple extension of the original proof of $DLB_{yi}(Q, S) \leq DTW(Q, S)$ given in [30].

Lemma 3 Let Q, S be two time series. It is necessary that $DLB_{yi}(Q, S) \leq ERP(Q, S)$.

Instead of just using $min(S)$ and $max(S)$, the lower bound proposed by Kim et al., denoted as $DLB_{kim}(Q, S)$, uses the first and last elements of Q and S as well, i.e., q_1, q_m and s_1, s_n. The lower bound is the maximum of: $|max(Q) - max(S)|$, $|min(Q) - min(S)|$, $|q_1 - s_1|$ and $|q_m - s_n|$. To adapt this lower bound for ERP, we need to take into account that for the aligned time series, the first or the last element may be a gap. With this simple change, we have the following result, which extends $DLB_{kim}(Q, S) \leq DTW(Q, S)$ shown in [18].

Lemma 4 Let Q, S be two time series. It is necessary that $DLB_{kim}(Q, S) \leq ERP(Q, S)$.

Finally, Keogh et al. introduced the concept of a bounding envelop for a time series Q, which is of the form: $Env(Q) = [(Min_1, Max_1), \ldots (Min_n, Max_n)]$. Min_i and Max_i represent the minimum and maximum value of the warping range of an element. The warping range of an element is the maximum number of times the element can be duplicated in local time shifting. To adapt the envelope to ERP, Min_i and Max_i are adjusted with the gap, exactly like in $DLB_{yi}(Q, S)$. The following extends: $DLB_{keogh}(Q, S) \leq DTW(Q, S)$ shown in [14].

Lemma 5 Let Q, S be two time series. It is necessary that $DLB_{keogh}(Q, S) \leq ERP(Q, S)$.

The effectiveness of the three adapted lower bounds will be empirically evaluated in the next section. The focus will be on the pruning power of these lower bounds. Notice that the lower bound DLB_{keogh} is

not directly indexable. Thus, in [31], Shasha et al. proposed a lower bound for DLB_{keogh}. We do not include this technique in our experimentation because we will make comparison with DLB_{keogh} directly. As a preview, we will show that our proposed technique dominates DLB_{keogh}, which translates to a superiority over the Shasha's lower bound of DLB_{keogh}.

Finally, as the comparisons are based on pruning power, we consider a sequential scan strategy for exploiting these lower bounds. This can be achieved by modifying procedure TrianglePruning in Figure 4 as follows:

- Delete lines 2 to 4 and line 8, as there is no longer necessary to keep *procArray*.
- Change line 1 to: $maxPruneDist = DLB_{yi}(Q,S)$, $DLB_{kim}(Q,S)$ or $DLB_{keogh}(Q,S)$.

4.2.2 A New Lower Bound for ERP

Below we develop a new lower bound specifically for ERP. Given a time series S of length n, we use the notation $sum(S)$ to denote $\sum_{i=1}^{n} s_i$. Then we define the distance $DLB_{erp}(Q,S)$ between Q of length m and S of length n as:

$$DLB_{erp}(Q,S) = |sum(Q) - sum(S)| \quad (8)$$

A key result here is the following theorem stating that this is indeed a lower bound for ERP.

Theorem 2 Let Q, S be two time series. It is necessary that $DLB_{erp}(Q,S) \leq ERP(Q,S)$.

Before we present a proof sketch of the theorem, we need the following results from the literature on a convex function. A function $f(x)$ is *convex* on an interval $[a,b]$ if for any two points x_1 and x_2 in $[a,b]$, $f[\frac{1}{2}(x_1+x_2)] \leq \frac{1}{2}[f(x_1)+f(x_2)]$.

Lemma 6 [23] Let $\lambda_1, \ldots, \lambda_K$ be non-negative real values such that $\sum_{i=1}^{K} \lambda_i = 1$. If f is a convex function on reals, then $f(\lambda_1 x_1 + \ldots + \lambda_K x_K) \leq \lambda_1 f(x_1) + \ldots + \lambda_K f(x_K)$, where $x_1, \ldots, x_K$ are real values.

As shown in [29], the following corollary can be established for Lp-norms which are convex functions on reals.

Corollary 1 For any sequence $S = [s_1, \ldots, s_K]$, and $1 \leq p < \infty$, we have: $K \cdot |mean(S)|^p \leq \sum_{i=1}^{K} |s_i|^p$.
Proof of corollary: Take $f(x) = |x|^p$, and $\lambda_i = 1/K$. Then $f(\sum_{i=1}^{K} \lambda_i s_i) = f((\sum_{i=1}^{K} s_i)/K) = |mean(S)|^p$. Thus, the inequality follows.

With the above corollary and Lemma 2, we can now establish Theorem 2.

Proof of Theorem 2: Given Q of length m and S of length n, let the aligned sequences, to give the best local time shifting outcome, be $\widetilde{Q}$ and $\widetilde{S}$ respectively. Both $\widetilde{Q}$ and $\widetilde{S}$ are of the same length, say K.

$$\begin{aligned}
ERP(Q,S) &= \sum_{i=1}^{K} |\widetilde{s}_i - \widetilde{q}_i| && \text{(Formula 7)} \\
&\geq K \cdot |mean(\widetilde{S} - \widetilde{Q})| && \text{(Corollary 1, } p=1\text{)} \\
&= K \cdot |\frac{\sum_{i=1}^{K} \widetilde{s}_i - \sum_{i=1}^{K} \widetilde{q}_i}{K}| \\
&= |\sum_{i=1}^{K} \widetilde{s}_i - \sum_{i=1}^{K} \widetilde{q}_i| \\
&= |\sum_{i=1}^{n} s_i - \sum_{j=1}^{m} q_j| && \text{(Lemma 2)} \\
&= DLB_{erp}(Q,S)
\end{aligned}$$

There are two points worth mentioning from the above theorem. The first point concerns the application of Lemma 2, i.e., $\sum_{i=1}^{K} \widetilde{s}_i = \sum_{i=1}^{n} s_i$. This is due to the fact that the gap element is $g = 0$. The beauty of this equality is that regardless of $\widetilde{S}$, which depends on Q, the quantity $sum(S)$ is unchanged and can be inserted into an index for lower bounding purposes for *any future* query Q. This leads to the second point – $sum(S)$ is a single value. Thus, only a 1-dimensional B+-tree is sufficient.

Figure 6 shows a skeleton of the algorithm for using the B+-tree for lower bounding with $sum(S)$. It first conducts a standard search for the value $sum(Q)$. This results in a leaf node I. The first k time series pointed to by I are used to initialize the *result* array. Then the leaf nodes of the tree are traversed using the pointer connecting a leaf node to its siblings. All the data values bigger than $sum(Q)$ are visited in ascending order. Similarly, all the data values smaller than $sum(Q)$ are visited in descending order. In both cases, if the lower bound distance $DLB_{erp}(Q,S)$ is smaller than the best k-NN distance so far, the true distance $ERP(Q,S)$ is computed and updates are made if necessary. Otherwise, the remaining data values can be skipped entirely.

4.3 Combining the Two Pruning Methods

For a non-metric distance function like DTW, lower bounding can be used, but pruning by the triangle inequality cannot. ERP is of course different. It is possible to combine both methods – use the triangle inequality to save the computation of the true distance $ERP(Q,S)$ *after lower bounding*. A skeleton is shown in Figure 7.

Recall from Figure 4 that in applying the TrianglePruning procedure, there are two unresolved issues: (i) the size of the pairwise distance matrix *pmatrix*; and (ii) the size of *procArray*, i.e., the maximum number of time series whose true ERP distances are

```
Procedure DLBerpk-NN(Q, k, Tree, result) {
  /* Tree ≡ a B+-tree storing sum(S) for all S in database */
  (1) sumQ = sum(Q)
  (2) conduct a standard B+-tree search on Tree using sumQ
        and let I be the leaf node the search ends up with
  (3) pick the first k time series pointed to by I and
        initialize result with the k true (sorted) ERP distances
  (4) let v_1, ..., v_h be the data values in all the leaf nodes
        larger than the data values visited in line 3. v_1, ..., v_h
        are sorted in ascending order.
  (5) for each v_i {
  (6)     bestSoFar = result[k].dist /* the k-NN distance so far */
  (7)     if ((v_i - sumQ) ≤ bestSoFar) /* need to check */
  (8)       for each S pointed to by the pointer with v_i {
  (9)         realDist = ERP(Q, S) /* compute true distance */
  (10)        if (realDist < bestSoFar) { /* update result */
  (11)            insert S and realDist into result,
                    sorted in ascending order of ERP distance
  (12)            bestSoFar = result[k].dist
            } /* end-if, line 10 */
        } /* end-for, line 8 */
  (13)  else break /* else, line 7, skip the rest */
      } /* end-for, line 5 */
  (14) repeat line 4 to 13 this time to the values w_1, ..., w_j
       in all the leaf nodes which are smaller than the data values
       visited in line 3. w_1, ..., w_j are sorted in descending
       order. Line 7 is modified to: if ((sumQ - w_i) ≤ bestSoFar)
  (15) return result
}
```

Figure 6: Algorithm for Applying DLB_{erp}

```
Procedure ERPCombineK-NN(Q, k, Tree, result) {
  Identical to Procedure DLBerpk-NN except:
    line 8 to 12 is replaced with:
    (8') for each S pointed to by the pointer with v_i
    (9')    invoke procedure TrianglePruning() in Figure 4
}
```

Figure 7: Algorithm for Applying first DLB_{erp} followed by the Triangle Inequality

kept for triangle inequality pruning. Below we resolve these issues to make procedure ERPCombineK-NN in Figure 7 practical for large databases and for limited buffer space situations.

Let *maxTriangle* denote the maximum number of time series whose true ERP distances are kept for triangle inequality pruning. This value should be determined at query time by the query engine. Hereafter we call these time series the *reference series*. There are two ways to pick these reference series:

- *static*: these reference series are randomly picked *a priori*. For that matter, they may not even be series contained in the database.
- *dynamic*: these reference series are picked as procedure ERPCombineK-NN runs. The choices are thus query dependent. In our implementation, we simply pick the first *maxTriangle* time series that fill up the fixed-size *procArray*.

For the static reference series strategies, the entire *pmatrix* is not needed. We only need the pairwise ERP distance matrix for each pair of reference series and data series. Thus, if N is the number of time series in the database, then the size of the this matrix is $N * maxTriangle$. For the dynamic reference series strategy, again the entire *pmatrix* is not needed. As the reference series are picked and kept, the appropriate column of the matrix is read into the buffer space. The buffer space requirement is *maxTriangle* columns, each of size N. Thus, the total buffer space required is again $N * maxTriangle$.

While the buffer space requirement is the same, there are tradeoffs between the two strategies. For the dynamic strategy, the advantage is that the reference series are chosen based on previously compared time series. Thus, there is no extra cost involved. The disadvantage is that at the beginning of running ERPCombineK-NN, there may not be enough reference series for the triangle inequality to be effective. It takes at least *maxTriangle* iterations to fill up *procArray*. In contrast, for the static strategy, all *maxTriangle* reference time series are available right at the beginning of running ERPCombineK-NN. The disadvantage is that the ERP distance must be computed between the query and each reference time series, which represents additional overhead. In the next section, we will present experimental results to compare these two strategies.

In closing, note that ERPCombineK-NN can be generalized to apply to any metric distance function – representing an extension to the GEMINI framework. That is, within the GEMINI framework, once lower bounding has been applied, we can add an additional step to apply the triangle inequality to eliminate more false positives, as long as the underlying distance function is a metric. Thus, in this paper, not only do we find a good marriage between distance functions, but we also develop a framework for combining two popular pruning strategies.

5 Experimental Evaluation

5.1 Experimental Setup

In this section, we present experimental results based on the 24 benchmark data sets used in [31, 14], the stock data set used in [26], and the random walk data set tested in [16, 14, 31]. All experiments were run on a Sun-Blade-1000 workstation with 1G memory under Solaris 2.8. All the algorithms listed in previous sections were implemented in C. The various lower bound strategies for DTW [30, 18, 14], OMNI-sequential and the HF algorithm for selecting reference series [11], were also implemented. We obtained the M-tree code from *http://www-db.deis.unibo.it/Mtree/*.

Our experiments measure either total time or pruning power. Total time includes both CPU and I/O, and is measured in seconds. Given a k-NN query Q, the pruning power is defined to be the fraction of the time series S in the data set for which the true distance $ERP(Q, S)$ is not computed (without introducing false negatives). Following [14, 31], we measure pruning power (P) because this is an indicator free of implementation bias (e.g., page size, thresholds). Moreover, we vary k from 1 to 5, 20, etc.

5.2 Lower Bounding vs Triangle Inequality

The first experiment addressed the issue of lower bounding versus the triangle inequality. To make the comparisons fair, sequential scans were used. Figure 8 shows the pruning power of $DLB_{yi}, DLB_{kim}, DLB_{keogh}, DLB_{erp}$ and the triangle inequality (denoted as TR) on the 24 benchmark data sets for $k = 1, 5, 20$. The pruning power shown is the average over 50 random queries. Each data set contains 200 time series with length 256. As a concrete example, we pick an arbitrary data set among the 24 and show the detailed pruning power figures below for $k = 20$. The third data set is picked and it contains the spot prices (foreign currency in dollars) over 10 years (10/9/86 to 8/9/96).

DLB_{kim}	DLB_{yi}	DLB_{keogh}	DLB_{erp}	TR
0.0	0.28	0.44	0.54	0.63

From the results shown in the above table and Figure 8, it is obvious that among the three previously proposed lower bounds, DLB_{keogh} performs uniformly the best. However, it is dominated by DLB_{erp} proposed in this paper. Furthermore, the triangle inequality (TR) consistently outperforms all the lower bounds. The larger the value of k, the larger is the difference. This shows the value of having a metric distance function.

5.3 Combining DLB_{erp} with Triangle Inequality

From Figure 8, the triangle inequality (TR) performs the best in pruning power. In Figure 9, TR is compared further with:

- M-tree (page size 16K). Compared to other metric space indexing structures, such as MVP-tree and Sa-tree, M-tree is designed to minimize both I/O cost and distance computations.
- OMNI-sequential (OMNI-seq). OMNI-seq is tested with different number of reference points that ranges from 5 to 100.
- B+-tree (page size 1K). This implements DLB_{erp} with an index structure (i.e., Algorithm 6).
- B+-tree incorporating both the triangle inequality and DLB_{erp} (i.e., Algorithm 7). Both the static and dynamic versions are included.

In the experiment, we found that the pruning power of OMNI-seq with 100 reference points behaves very similar to that of TR. This is because they use the same strategies to remove false alarms; the only difference is the number of reference points that they use. Therefore, we do not include the results of OMNI-seq in Figure 9.

From the figure, the first observation is that all the methods do well when $k = 1$. But with $k = 20$, the separation is clear. M-tree is by far the worst, even dominated by the triangle inequality based on sequential scan. This is because with M-tree, the algorithm uses the distance information below each scanned node, rather than on all the true distances previously computed. This restricts the effectiveness of the pruning.

The second observation is that while TR (based on sequential scan) outperforms DLB_{erp} with sequential scan, the reverse is true when DLB_{erp} is used with a B+-tree. This shows the effectiveness of the B+-tree index in facilitating lower bound pruning.

The third observation is that the combination approaches B+TR(static) and B+TR(dynamic) are the best – better than B+-tree alone or the triangle inequality alone. This shows the beauty of combining both lower bounding and pruning by the triangle inequality. For both the static and dynamic combination approaches, the number of reference series varies from 5 to 100. The graph only shows the result with 100 reference series. The following table is included to show the pruning power in greater details on the third data set with $k = 20$.

M-tree	TR	B+tree	B+TR(static)		
			20	50	100
0.59	0.63	0.72	0.76	0.80	0.82

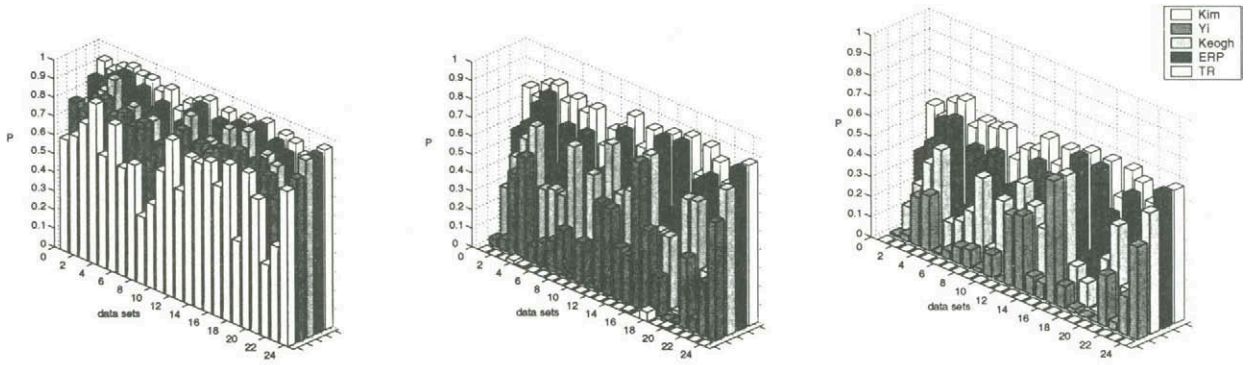
B+TR(dynamic)			B+HF	OMNI-seq
20	50	100	(100)	(100)
0.74	0.76	0.77	0.76	0.63

For both the static and dynamic combination approaches, increasing the number of reference series improves the pruning power as expected. And the static approach appears to perform better than the dynamic approach. However, as will be shown in later experiments, this perspective on pruning power is not yet the complete picture.

Note that in the above table, we also include the pruning power results using the HF algorithm to select the reference series and the OMNI-sequential with 100 reference points [11]. Both are dominated by the static and dynamic approaches we developed.

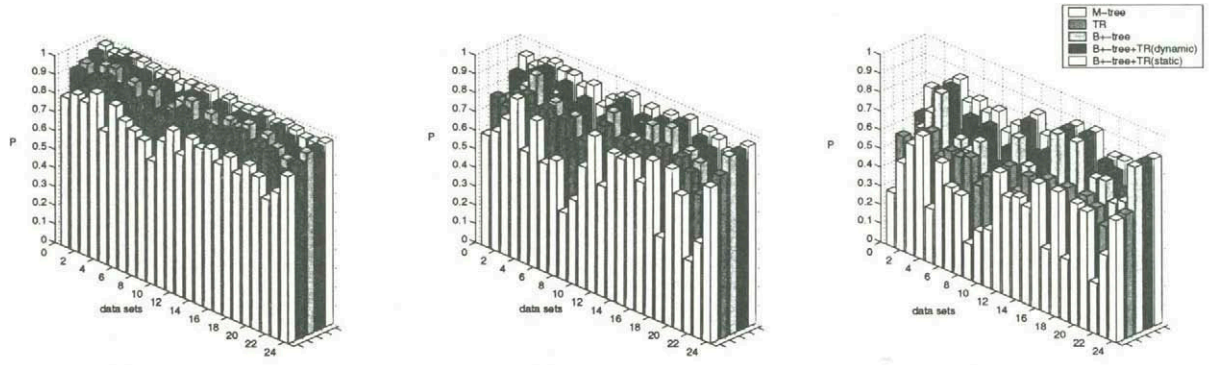
5.4 Total Time Comparisons

So far we have presented the pruning power results. However, these results do not represent the complete picture because higher pruning power may require additional time spent on pre-processing. Figure 10 shows the total time spent on answering the k-NN queries. This time includes the pre-processing time and the query processing time (both CPU and I/O included). The time taken by TR (the triangle inequality alone) is not shown in the figure because it takes much longer than the others. We also found that the time cost of OMNI-seq is very similar to that of B+TR(static)), because both of them spend the same amount of time on computing the distance between query time series and the reference time series. Thus, we did not include the results of OMNI-seq in Figure 10. Again,



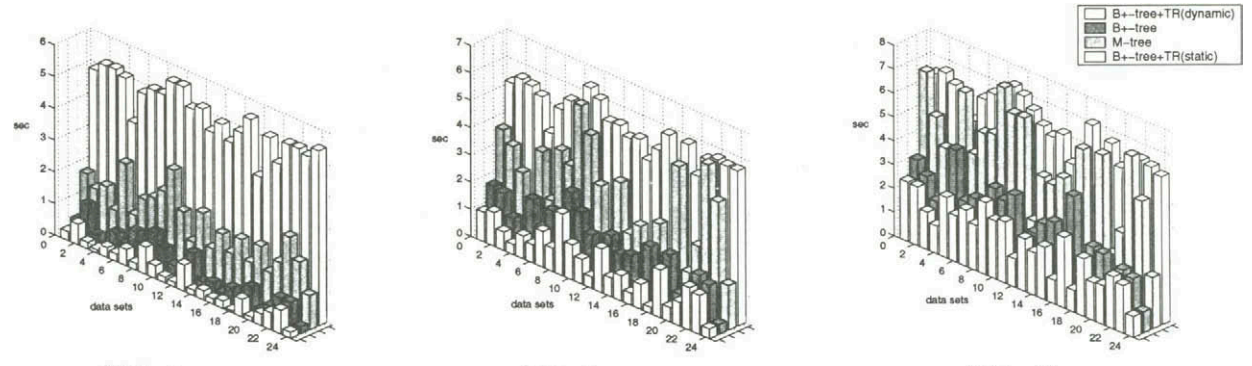
(a) k=1 (b) k=5 (c) k=20

Figure 8: Lower Bounding vs the Triangle Inequality: Pruning Power



(a) k=1 (b) k=5 (c) k=20

Figure 9: Lower Bounding Together with the Triangle Inequality: Pruning Power



(a) k=1 (b) k=5 (c) k=20

Figure 10: Total Time comparisons: 24 Benchmark Data Sets

the graphs only show the results of combined methods with 100 reference sequences. The detailed run time (in seconds) on the third data set with $k = 20$ and 100 reference series is shown below.

M-tree	B+tree	B+TR (static)	B+TR (dynamic)	OMNI-seq
3.89	1.84	6.36	1.58	5.68

Among the approaches shown in the figure and table, the static combination approach (i.e., B+TR(static)) takes the most time, followed by OMNI-seq, M-tree and the B+-tree alone. The dynamic combination approach appears to be the best, in delivering very good pruning power but not at the expense of sacrificing pre-processing time.

However, it would be a mistake to conclude at this point that the dynamic combination approach is the clear winner. The main reason is that the 24 benchmarks are all small data sets, none of which contains more than 200 time series. We use them because they have been widely used as benchmarks, and represent a wide spectrum of applications and data characteristics. However, to address the size issue, the following set of results is based on much larger data sets.

5.5 Scalability Tests

We used two data sets. The first data set was a real stock data set from [26], which contains daily stock prices from 4622 American companies. Each time series contains 1024 daily closing prices, starting from March 1995 to March 1999. To make the data set bigger, we segmented each time series into 4 disjoint series with length 256 each. Thus, the total number of time series is 18,488. The other data set was the random walk data set used in [16, 14, 31]. We generated 65,536 and 100,000 random walk time series, each of length 256.

Figure 11 shows the total time taken to answer k-NN queries. The time taken for five methods are included: M-tree, B+-tree alone, B+TR(static,400), B+TR(dynamic,400) and OMNI-sequential(400), where 400 refers to the number of reference time series. It is clear that the static and dynamic approaches perform the best. In fact, the static approach manages to catch up with the dynamic approach, and in some cases, it even outperforms. The reason is that when the data set is large and selectivity is high ($k > 1$), the pre-processing represents a small overhead relative to the search. The improvement in pruning power can then compensate for the overhead. We also ran the experiments to confirm the pruning power on these large data sets. Same conclusions can be drawn as for the 24 benchmarks.

6 Conclusions and Future Work

In this paper, we present the ERP distance function for time series similarity retrieval. ERP can be viewed as a perfect marriage between L1-norm and edit distance. It resembles L1-norm in being a metric distance function, and it resembles edit distance in being able to handle local time shifting. We show that this new distance function is as natural for time series as existing distance functions such as DTW and LCSS.

Because ERP is a metric distance function, the triangle inequality can be used for pruning for k-NN queries. Furthermore, we develop a new lower bound for ERP distance. Unlike existing lower bounds which require indexing with a multi-dimensional index, the proposed lower bound is 1-dimensional and can simply be implemented in a B+-tree, which reduces the storage requirement and save the possible I/O cost in consequence. Because pruning by the triangle inequality and by lower bounds are orthogonal, we combine the two strategies. This combination can be considered as an extension of GEMINI framework for indexing time series data with a metric.

We conducted extensive experimentation using 24 benchmarks and other large data sets. Consistently, our new lower bound dominates existing strategies. More importantly, the combination approaches which incorporate both lower bounding and pruning by the triangle inequality deliver superb pruning power as well as wall clock time performance. Among the two combination approaches, both the static and the dynamic approach are viable alternatives depending on k and the size of the database.

A direct extension of the similarity-based time series retrieval is the search of trajectories of moving objects in multimedia databases. Unlike the time series data that we used in this paper, trajectories of moving objects are multi-dimensional and contain possible noises, which brings new challenges. Our future research directions will include the work in this direction.

Acknowledgements: Thanks to Eamonn Keogh, Michalis Vlachos, X. Sean Wang for providing data sets.

References

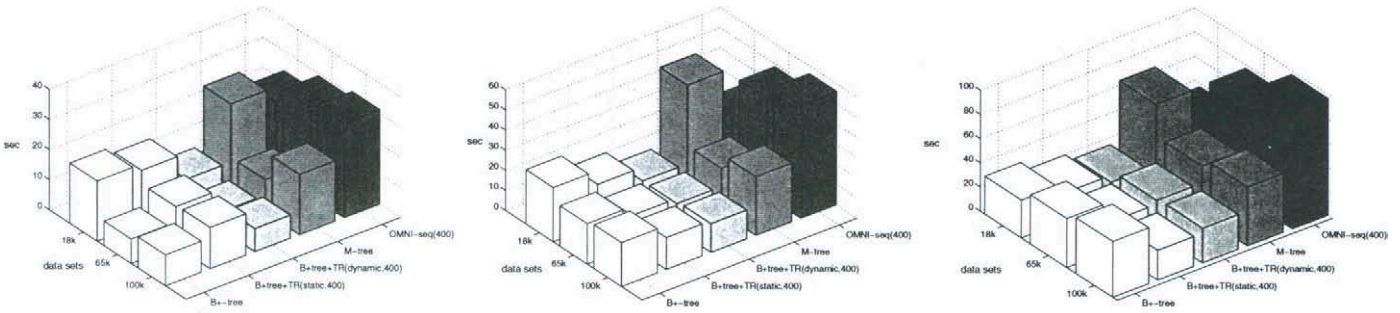
[1] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *Proc. 4th Int. Conf. of Foundations of Data Organization and Algorithms*, pages 69–84, 1993.

[2] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zaït. Querying shapes of histories. In *Proc. 21th Int. Conf. on Very Large Data Bases*, pages 502–514, 1995.

[3] D. J. Berndt and J. Clifford. Finding patterns in time series: A dynamic programming approach. In *Advances in Knowledge Discovery and Data Mining*, pages 229–248, 1996.

[4] J. S. Boreczky and L. A. Rowe. Comparison of video shot boundary detection techniques. In *Proc. 8th Int. Symp. on Storage and Retrieval for Image and Video Databases*, pages 170–179, 1996.

[5] T. Bozkaya and M. Ozsoyoglu. Indexing large metric spaces for similarity search queries. *ACM Trans. Database Sys.*, 24(3):361–404, 1999.



(a) k=1 (b) k=5 (c) k=20

Figure 11: Total Time Comparisons: Large Data Sets

[6] L. Chen, M. T. Özsu, and V. Oria. Robust and efficient similarity search for moving object trajectories. In *CS Tech. Report. CS-2003-30, School of Computer Science, University of Waterloo.*

[7] E. Chvez, G. Navarro, R. Baeza-Yates, and J. L. Marroquń. Searching in metric spaces. *ACM Comput. Surv.*, 33(3):273–321, 2001.

[8] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proc. 23th Int. Conf. on Very Large Data Bases*, pages 426–435, 1997.

[9] G. Das, D. Gunopulos, and H. Mannila. Finding similar time series. In *Proc. 1st European Symp. on Principles of Data Mining and Knowledge Discovery*, pages 88–100, 1997.

[10] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 419–429, 1994.

[11] R. F. S. Filho, A. J. M. Traina, C. Traina Jr., and C. Faloutsos. Similarity search without tears: The OMNI family of all-purpose access methods. In *Proc. 17th Int. Conf. on Data Engineering*, pages 623–630, 2001.

[12] P. Geurts. Pattern extraction for time series classification. In *Proc. 5th Int. Conf. on Principles and Practice of Knowledge Discovery in Databases*, pages 115–127, 2001.

[13] D.Q. Goldin and P.C. Kanellakis. On similarity queries for time series data: Constraint specification and implementation. In *Proc. of the Int. Conf. on Principles and Pratice of Constraint Programming*, pages 23–35, 1995.

[14] E. Keogh. Exact indexing of dynamic time warping. In *Proc. 28th Int. Conf. on Very Large Data Bases*, pages 406–417, 2002.

[15] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2000.

[16] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 151–162, 2001.

[17] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. In *Proc. 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 102–111, 2002.

[18] S Kim, S. Park, and W. Chu. An indexed-based approach for similarity search supporting time warping in large sequence databases. In *Proc. 17th Int. Conf. on Data Engineering*, pages 607–614, 2001.

[19] F. Korn, H. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 289–300, 1997.

[20] K.P.Chan and A.W-C Fu. Efficient time series matching by wavelets. In *Proc. 15th Int. Conf. on Data Engineering*, pages 126–133, 1999.

[21] G. Navarro. Searching in metric spaces by spatial approximation. *The VLDB Journal*, 11:28–46, 2002.

[22] I. Popivanov and R. J. Miller. Similarity search over time series data using wavelets. In *Proc. 17th Int. Conf. on Data Engineering*, pages 212–221, 2001.

[23] M. H. Protter and C. B. Morrey. *A First Course in Real Analysis.* Sprinter-Verlag, 1977.

[24] E. Vidal and F. Casacuberta. On the verification of triangle inequality by dynamic time-warping dissimilarity measures. *Speech Commun.*, 7(1):67–79, 1988.

[25] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *Proc. 18th Int. Conf. on Data Engineering*, pages 673 - 684, 2002.

[26] C.Z. Wang and X. Wang. Supporting content-based searches on time series via approximation. In *Proc. 12th Int. Conf. on Scientific and Statistical Database Management*, pages 69–81, 2000.

[27] M. S. Waterman, T. F. Smith, and W. A. Beyer. Some biological sequence metrics. advances in mathematics. *Advances in Mathematics*, (20):367–387, 1976.

[28] R. Weber, H.-J Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proc. 24th Int. Conf. on Very Large Data Bases*, pages 194–205, 1998.

[29] B-K Yi and C. Faloutsos. Fast time sequence indexing for arbitrary Lp norms. In *Proc. 26th Int. Conf. on Very Large Data Bases*, pages 385–394, 2000.

[30] B-K Yi, H. Jagadish, and C. Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proc. 14th Int. Conf. on Data Engineering*, pages 23–27, 1998.

[31] Y. Zhu and D. Shasha. Warping indexes with envelope transforms for query by humming. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 181–192, 2003.