A Parallel, Structural Comparison Scheme of Time-Series Implemented on a PC Cluster

This paper presents a scheme for parallel computation of similarity matrix using a PC cluster. Pairwise comparison of N sequences essentially requires \( N(N + 1)/2 \) computational tasks; therefore, a simple and efficient comparison method such as Euclidean distance or Dynamic Time Warping is often used. Focusing on the fact that the pair-wise comparison can be processed independently for different pairs of sequences, we employ a parallel computation scheme with a PC cluster and attempt to include more complex, structure-based pairwise comparison method. Experimental results on the cylinder-bell-funnel data set demonstrate that the combination of parallel computation and structural comparison may be used for handling a difficult clustering task of time series.

A Parallel, Structural Comparison Scheme of Time-series Implemented on a PC Cluster

Shoji Hirano  Shusaku Tsumoto
Department of Medical Informatics, Shimane University School of Medicine
89-1 Enya-cho, Izumo, Shimane 693-8501, Japan
hirano@ieee.org, tsumoto@computer.org

Abstract

This paper presents a scheme for parallel computation of similarity matrix using a PC cluster. Pairwise comparison of N sequences essentially requires \( N(N+1)/2 \) computational tasks; therefore, a simple and efficient comparison method such as Euclidean distance or Dynamic Time Warping is often used. Focusing on the fact that the pairwise comparison can be processed independently for different pairs of sequences, we employ a parallel computation scheme with a PC cluster and attempt to include more complex, structure-based pairwise comparison method. Experimental results on the cylinder-bell-funnel data set demonstrate that the combination of parallel computation and structural comparison may be used for handling a difficult clustering task of time-series, which may be achieved with the simple comparison methods.

2. Parallel Computation of Dissimilarities

2.1. PC cluster

The cluster system consists of eight PCs (DELL PowerEdge 1750). Each PC is equipped with dual Intel Xeon processors (2.4GHz) and 1GB Memory. The operating system and message-passing libraries were RedHat Linux 9 and MPICH 1.2.5, respectively. Since a Xeon processor is recognized as two processing units, the system contains 32 processing elements (PEs) in total. Figure 1 shows the architecture of our cluster. PCs were connected each other via 1000Base-T TCP/IP network. Since inter-communications between nodes were small in our applications, all communications from/to one node shared one GbE connection. All of the eight nodes perform computation as slave nodes and node 1 additionally behaves as a master node for running the program. The performance measured using Himeno Benchmark (size M) under 32 PEs was about 5.5 GFLOPS.

2.2. Computational scheme

Let us first denote a set of PEs by \( PE = \{PE_1, PE_2, \ldots, PE_{N_{PE}}\} \), where \( N_{PE} \) denotes the...
number of PEs (=32 in our system). Besides, let us denote a set of possible pairs of \( N \) objects by \( P = \{\{1,1\},\{1,2\},\ldots,\{N,N\}\} \) and the number of pairs in \( P \) by \( N_{pair} \). Here we include the pairs of the same objects, i.e. \{1,1\} for comprehensibility. We assume that the matrix is symmetric, and omit the symmetric elements. Consequently, the number of elements is \( N(N+1)/2 \).

Figure 2 illustrates the computational scheme. The dataset is stored in advance on the shared storage. Firstly, \( PE_1 \) reads the dataset, and set the index of pairs \( i \) to be compared by \( PE_1 \) to \( i_1 = 1 \) (=pair \( \{1,1\} \)). Secondly, \( PE_1 \) performs multiscale matching for pair \( i_1 \). The code for Multiscale Matching is implemented to input two time-series and output their dissimilarity, thus it returns the dissimilarity of two sequences in \( i \). The dissimilarity is then stored on the shared storage, along with the ID of the PE, and the IDs of the two sequences. Thirdly, \( PE_1 \) increases the index \( i_1 \) to \( i_1 + N_{PE} \), in order to proceed to the next pair. Pairs from \( i_2 \) to \( i_1 + N_{PE} - 1 \) are skipped because they are being processed by other PEs. If the increased index \( i_1 \) exceeds the total number of pairs, \( PE_1 \) will terminate the process, otherwise repeat from MSM for this \( i_1 \).

The above process can proceed in \( N_{PE} \) parallel, since the dissimilarity of a pair can be determined independently to the other pairs.

3. Structural Comparison of Time-Series by Modified Multiscale Matching

Multiscale matching (MSM) [3] was originally developed as a method for comparing two planar curves by partly changing observation scales. It divides a contour of the object into partial contours based on the place of inflection points. After generating partial contours at various scales for each of the two curves to be compared, it finds the best pairs of partial contours that minimize the accumulated dissimilarity while preserving completeness of the concatenated contours. We have extended this method so that it can be applied to the comparison of two one-dimensional temporal sequences. Figure 3 provides an illustrative explanation of multiscale time-series matching. Due to space limitation, we omit the basics; for details see [5].

In order to elude the problem of shrinkage at high scales noted in [6], we have designed a new matching scheme for temporal sequences. The key points are:

- Multiscale shapes are used only to obtain the hierarchy of the segments.
- Segment difference and replacement cost are directly obtained from segments at the lowest scale.

Figure 4 left provides new shape parameters for single segment \( a_i \). Since we use only segments at the lowest scale, we omit \( (0) \) of \( a_i \) for simplicity. Features of \( a_i \) consist of the following four components:

1. Amplitude \( a_i \): vertical amplitude measured from the baseline of the segment to peak point \( p_k \). Baseline is a straight line connecting both ends of the segment.
2. Width \( w_i \): horizontal width of the segment.
3. Height \( h_i \): vertical shift of the segment w.r.t. both ends.
4. Phase \( p_i \): phase of the segment; measured from the starting point of the entire sequence to the starting point of the segment.

Figure 4 center illustrates our shape parameters for merged segments. Suppose that \( n \) contiguous segments
\((a_i, a_{i+1}, \ldots, a_{i+n-1})\) can be replaced into one segment \(a_m^{(k)}\) at scale \(k\) according to the segment hierarchy. Then we generate the new shape parameters for the replaced segment \(a_m^{(k)}\) as follows: (1) determine the new peak point to the centroid \(pk[a_i, a_{i+1}, \ldots, a_{i+n-1}]\) of all \(n\) peaks. (2) using \(pk[a_i, a_{i+1}, \ldots, a_{i+n-1}]\) and both ends of the merged segment, calculate the four parameters \(a(m)^{(k)}\), \(w(a_m^{(k)})\), \(h(a_m^{(k)})\), \(p(a_m^{(k)})\). Note that we do not directly use the shape of in order to obtain the above four parameters. Figure 4 right illustrates the problem of shrinkage. Since we have multiscale description, we can obtain the shape of \(a_m^{(k)}\) at scale \(k\) and directly calculate \(a(a_m^{(k)})\) as a single segment. However, it will shrink toward the baseline as \(k\) becomes large, distorting the amplitude value (actually, all of the four parameters are subject to be distorted by shrinkage).

Now the segment difference is defined as:

\[
d(a_m^{(k)}, l_1^{(k)}) = \max(d_a, d_w, d_h, d_p) + \gamma (\text{cost}(a_m^{(k)}) + \text{cost}(l_1^{(k)}))
\]

where \(d_a = d_a(a_m^{(k)}, l_1^{(k)})\) denotes difference of amplitude between segments \(a_m^{(k)}\) and \(l_1^{(k)}\), and other symbols \(d_a, d_w, d_h, d_p\) are interpreted analogously. \(\text{cost}(a_m^{(k)})\) is a cost function for replacement. For \(a_m^{(k)}\) that replaces \((a_i, a_{i+1}, \ldots, a_{i+n-1})\), cost is defined by

\[
\text{cost}(a_m^{(k)}) = \begin{cases} 
0, & \text{if } n = 1 \\
\min \left\{ \frac{1}{a_m^{(k)}} \sum_{a=1}^{i+n-1} |a(a_m^{(k)})| \right\}, & \text{otherwise.}
\end{cases}
\]

This cost becomes low when the replacement enlarges average amplitude.

Based on the above dissimilarity measure \(d(a_m^{(k)}, l_1^{(k)})\), the best set of segment pairs \(P = \{p_w, 1 \leq w \leq N_p | p_1 = ((a_1-a_i, b_1-b_1)), p_2 = (a_{i+1}+a_j, b_{i+1}+b_m), \ldots, p_{N_p} = (a_{k+n-1}+a_N, b_{n+1}+b_M)\}\) that minimizes the accumulated difference \(D(A, B)\) between sequences \(A\) and \(B\),

\[
D(A, B) = \sum_{i=1}^{N_p} d(a_p^i, b_p^i)
\]
is searched throughout all scales.

It is important to evaluate multiple factors in comparing two shapes to achieve good matching results. For example, our dissimilarity measure evaluates four factors: amplitude, width, phase, and height, and it produces intuitively good matching results that fit with the human perception. However, since the local dissimilarity is obtained as a maximum of these four components, and what each factor evaluates is essentially different, it is better not to directly use \(D(A, B)\) as the resultant dissimilarity of sequences \(A\) and \(B\). Therefore, we employ the following strategy in determining the final difference of two sequences to be outputted.

1. Find the best set of segment pairs \(P\) according to the local dissimilarity measure \(d(a_m^{(k)}, l_1^{(k)})\), consisting of the four components: \(a, w, p, h\).

2. For each pair \((a_p^i, b_p^i)\) in \(P\), we further calculate the accumulate difference of height, \(d_h\), by

\[
D_h(A, B) = \sum_{i=1}^{N_p} d_h(a_p^i, b_p^i).
\]

This enable us to uniform the dimension of the dissimilarity with that of signal height, with better linearity and understandability as the dissimilarity measure.

### 4. Experimental Results

We examined the usefulness of the proposed method on the cylinder-bell-funnel data set [7] [8]: a simple synthetic data set which is well-known and frequently used in the temporal data mining community. Experiments were performed as follows. (1) Generate a data set containing a total of 384 sequences; 128 sequences for each of the three classes, cylinder, bell, and funnel. (2) Compute the dissimilarities for all pairs of sequences in the data sets using the proposed method. This produced a 384 × 384 symmetric dissimilarity matrix. (3) Remove one sequence, and predict its class according to the class label of the nearest sequence. The nearest sequence is selected according to the dissimilarity matrix. (4) Repeat step (3) for each of the 384 sequences, and we have the prediction error rate. Namely, we performed the leave-one-out validation with 1-Nearest Neighbor classification algorithm, using the dissimilarity matrix obtained by the proposed method in [9].

Before applying MSM, all of the input sequences were normalized in both horizontal and vertical directions by dividing by their standard deviation (because the length of sequences in cylinder-bell-funnel dataset were all the same, we simply normalized them in horizontal direction by dividing their length). The parameters in MSM were set as follows: starting scale = 1.0, scale interval = 0.5, number of scales = 100, cost weighting \(\gamma = 0.5\).

The error rate was 0.054, which is quite better compared to the results summarized in [9] (2nd best below the Euclidean distance, whose error rate = 0.003; we also reproduced the same result).

Next, we evaluated whether the dissimilarity matrix can be used to form meaningful clusters. We modified parts (3)-(4) of the above experimental procedures as follows. (3) remove one sequence, and using the 383 × 383 matrix, perform conventional average-linkage agglomerative clustering [10] specifying the number of clusters to 3. (4) assign
class label to each of the three clusters by the voting. (5) Perform 1-Nearest Neighbor classification for the removed sequence, and evaluate the classification accuracy. (6) Remove another sequence and perform the same procedure. This is applied to all the 384 sequences.

The error rate was 0.042, similar to the previous experiment. We also performed the same experiments using the Euclidean distance, and its error rate was 0.216. This relatively high error rate of Euclidean distance implied that the dissimilarity matrix failed to form the clusters representing correct class distributions. Figure 5 provides a dendrogram of the entire dataset generated in combination with the MSM-based dissimilarity matrix and conventional average-linkage agglomerative clustering method (same as above procedure). It can be seen that the three-class structure of this data set was correctly represented. By setting an appropriate cutting point on the dendrogram, we can obtain three clusters: cluster 1 (133 seqs; 128 funnel, 2 cylinder, 3 bell), 2 (125 seqs; 125 cylinder) and 3 (126 seqs; 126 bell). Figure 6 shows examples of sequences in the three clusters. Although several sequences belonged to cylinder or bell were miss-clustered into the 'funnel' cluster, the high accuracy (> 98.7%) and highly reproductive accuracy (leave-one-out, >95.8%) demonstrate the high performance of the proposed method in terms of accuracy.

It took about 140 minutes to construct MSM-based dissimilarity matrix for CBF data using the parallel scheme. A PE firstly finished its calculation in about 127 minutes. The ratio of active PEs decreased to 70% within the next 3 minutes. Then it gradually decreased to 50% in about 6 minutes and steeply decreased to 90% in 2 mins. A few PEs remained active until the end. This may suggest a need for better load balancing, considering that the complexity of each pairwise comparison may change depending on the number of possible combinations of segments.

5. Conclusions

This paper has presented a parallel scheme for computing a dissimilarity matrix using a PC cluster. An MSM-based dissimilarity computation system has been implemented. The experimental results demonstrated that the use of rather complex, structural similarity computation method lead to the successful clustering of CBF dataset. Although it revealed a difficulty in load balancing, the parallel scheme on a PC cluster could greatly reduce the computation time. In the future we try to perform quantitative performance investigation of the system.

Acknowledgment

This work was supported in part by the Grant-in-Aid for Scientific Research on Priority Area (2)(#13131208) by the Ministry of Education, Culture, Science and Technology of Japan.

References