

Financial Time Series Indexing Based on Low Resolution Clustering

Tak-chung Fu^{1,2,‡}, Fu-lai Chung¹, Robert Luk¹ and Chak-man Ng²

¹Department of Computing
Hong Kong Polytechnic University
Hungghom, Kowloon, Hong Kong.
{cstcfu, cskchung, csrluk}@comp.polyu.edu.hk

²Department of Computing and Information Management
Hong Kong Institute of Vocational Education (Chai Wan)
30, Shing Tai Road, Chai Wan, Hong Kong.
cmng@vtc.edu.hk

Abstract

One of the major tasks in time series database application is time series query. Time series data is always exist in large data size and high dimensionality. However, different from traditional data, it is impossible to index the time series in traditional database system. Moreover, time series with different lengths always coexists in the same database. Therefore, development of a time series indexing approach is of fundamental importance for maintaining an acceptable speed for time series query. By identifying the perceptually important points (PIPs) from the time domain, time series of different lengths can be compared and the dimensionality of the time series can be greatly reduced. In this paper, a time series indexing approach, based on clustering the time series data in low resolution, is proposed. This approach is customized for stock time series to cater for its unique behaviors. It follows the time domain approach to carry out the indexing process which is intuitive to ordinary data analysts. One may find it particularly attractive in applications like stock data analysis. The proposed approach is efficient and effective as well. As demonstrated by the experiments, the proposed approach speeds up the time series query process while it also guarantees no false dismissals. In addition, the proposed approach can handle the problem of updating new entries to the database without any difficulty.

1. Introduction

Recently, time series data is of growing importance in many new database applications, such as data mining or data warehousing. A time series is a sequence of real numbers, each number representing a value at a time point. For example, the sequence could represent sales, exchange rates, weather data, biomedical measurements, stock prices ... etc.

We are often interested in finding similar time series [1] and querying time series database [2]. A time series database stores a large number of time series. The way of locating time series of interest in massive time series data efficiently is an important and non-trivial problem. Thus, indexing those time series is one of the possible solutions for increasing or at least maintaining an acceptable speed.

In this paper, an approach for indexing stock time series is proposed. A new time series representation scheme for various stock data analyses and mining tasks is first introduced. Stock time series has its own characteristics over other time series data such as the data from scientific areas like electrocardiogram ECG. For example, a stock time series is typically characterized by a few critical points and multi-resolution consideration is always necessary for long-term and short-term analyses. In addition, technical analysis is usually used to identify patterns of market behavior, which have high probability to repeat themselves. These patterns are similar in the overall shape but with different amplitude and/or duration. The perceptually important point (PIP) identification technique is adopted to determine data point importance and transform the original time series data into lower resolution. In addition, the time series database is indexed by clustering technique based on the low resolution of the time series. The paper is organized into six sections. Next section contains a discussion of related works. In section 3, the PIP time series representation framework based on data point reordering is introduced. The proposed time series indexing approach based on clustering is in Section 4. The simulation results are reported in Section 5 and the final section makes the conclusions of the paper.

2. Related Work

Reference [2] proposed the problem of similarity search in time series database and this problem has become an active research area afterward. Moreover, the problem of

‡ Corresponding author

indexing time series has attracted many researchers in the database community. The original GEMINI framework with different dimensionality reduction approaches in reference [3] was applied to time series indexing. Furthermore, most of the techniques for signal analysis require the data which is in the frequency domain. Therefore, distance-preserving orthonormal transformations are often used to transform the data from the time domain to the frequency domain. Usually, data-independent transformation is applied to where the transformation matrix is determined by prior and independent input data. One of the most popular data-independent transformations is the Discrete Fourier transform (DFT) [2, 4, 5].

In reference [6], it identifies a safe set of Fourier transformations and presents a query processing algorithm that uses the underlying multidimensional index built over the data set to answer similarity queries efficiently. On the other hand, the similarity search methods perform dimensionality reduction on the data, and then use a multidimensional structure to index the data in the transformed space. The technique was introduced in reference [2] and extended in references [4, 5]. Since the distance between two signals in the time domain is the same as their Euclidean distance in the frequency domain, the DFT performs well in preserving essentials in the first few coefficients. For efficient access, a multidimensional index is constructed using the first few Fourier coefficients. When a similarity query is submitted, the index is used to retrieve the sequences that are at a certain small distance away from the query sequence.

Moreover, there are several kinds of indexing technique based on the frequency domain: Wavelets, Discrete Wavelet Transform (DWT) [7, 8, 9, 10], Singular Value Decomposition (SVD) [11] and Inner Products [12]. It can be concluded that traditional query process for time series data transforms data from the time domain into the frequency domain. However, it is less controllable by the end users and also affects the visualization ability. Therefore, this is not well suited for pattern queries that find time series patterns with many dynamical and specified user constraints such as stock time series queries.

On the other hand, the time domain approach works on the temporal data points directly, for example, Piecewise Aggregate Approximation (PAA) [13] and Adaptive Piecewise Constant Approximation (APCA) [14]. The majority of works focused solely on performance issues in the time domain, though some authors also considered other issues such as supporting non Euclidean distance measures [5, 13, 14] and allowing queries time series of different lengths [13, 14, 15]. There is another popular dissimilarity metric called the “time warping” distance. From the indexing viewpoint, this metric presents two major challenges: (a) it does not lead to any natural indexing “features”, and (b) it requires time quadratic in the sequence

length when comparing two sequences. To address each problem, reference [16] proposes to map sequences into points, with little compromise of “recall” for speeding up the query process. Moreover, reference [17] introduces a technique for the exact indexing of dynamic time warping. The same author gives a comprehensive survey on time series data mining in paper [18].

Furthermore, reference [15] presents a method to search time series data which first transforms time sequences into symbol strings using changeable ratio between contiguous data points in time series. Next, a suffix tree is built to index all suffixes of the symbol strings. The focus of this paper is to demonstrate how this method can adapt to the processing of the dynamically constrained time series queries. Recent work in reference [19] proposed a new symbolic representation and the transformed time series adopts the same indexing technique as reference [15].

There are much more time series indexing techniques [20, 21, 22, 23, 24] which address the problem of similarity search in large time series databases efficiently. To sum up, much of the recent database technology research on this problem, or similar ones, can be characterized procedurally in the following general manner [25]: (i) To find an approximation and robust representation for a time series, for example, Fourier coefficients, piecewise linear models, etc.; (ii) To define a flexible indexing function that can handle various pattern deformations (scaling, transformations, etc.); and (iii) To provide an efficient scalable and updatable algorithm by using the adopted representation and matching function for massive time series data sets. In this paper we proposed a time series indexing framework based on data point importance for dimensional reduction. It is found especially useful when focusing on time series domain that the fluctuations of the time series are important signals and multi-resolution consideration is necessary (i.e. technical analysis in financial domain).

3. Reordering of Time Series Data Points

In this section, the proposed time series representation scheme is introduced. It is based on the reordering of the data points in the time series. Instead of storing the time series data according to time, or transforming it into other domains (e.g. frequency domain), data points of a time series are stored according to their importance. Based on this representation, the time series can transform to low resolution for indexing. Two main steps are needed for reordering the time series: perceptually important point identification and data point importance list construction. They are described in the following subsections. Table I shows most of the notations used in this paper.

Table I. Notations used in this paper

Symbol	Meaning
P	A time series $P=p_1, \dots, p_m$
Q	A query pattern $Q=q_1, \dots, q_n$
SP	Low resolution from P , $SP=sp_1, \dots, sp_n$ where $n \leq m$
L	The data point importance list $L=l_1, \dots, l_m$
k	Number of cluster
M_i	Cluster centroid $M=m_1, \dots, m_n$ where $i=1$ to k
N	Number of object in the database, x_1, \dots, x_N

3.1 Perceptually Important Point Identification

In view of the importance of extreme points in stock time series, the identification of perceptually important points (PIP) is first introduced in [26] and used for pattern matching of technical (analysis) patterns in financial applications. The idea was later found similar to a technique proposed about 30 years ago for reducing the number of points required to represent a line [27] (see also [28]). We also found independent works in [29, 30, 31] which work on similar ideas. In this subsection, let us review this idea by revisiting our previously proposed PIP identification process.

The frequently used stock patterns are typically characterized by a few critical points. For example, the head-and-shoulder pattern consists of a head point, two shoulder points and a pair of neck points. These points are perceptually important in the human identification process and should be considered of higher importance (Fig.1).

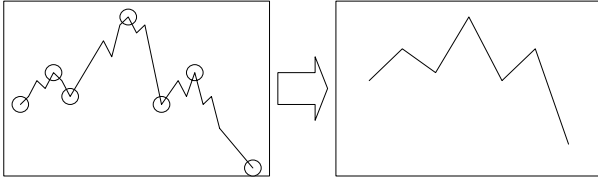


Figure 1. PIP identification results (from 30 data points in the data sequence P to 7 PIPs)

The PIP identification scheme follows this idea by locating those perceptually important points in the data sequence P in accordance with the query sequence Q (for pattern matching applications) or the specified number of PIPs. The locating process works as follows.

With sequences P and Q being normalized (for shifting and uniform amplitude scaling invariant), the PIPs are located in order according to Fig.2. Currently, the first two PIPs will be the first and the last points of P . The next PIP will be the point in P with maximum distance to the first two PIPs. The fourth PIP will then be the point in P with maximum distance to its two adjacent PIPs, either in between the first and second PIPs or in between the second and the last PIPs. The PIP location process continues until its number is equal to the length of query sequence Q or the required number of

PIPs is reached.

```

Function Point_locate (P,Q)
  Input:  sequence P[1..m], length of Q[1..n]
  Output: pattern SP[1..n]
Begin
  Set sp[1]=p[1], sp[n]=p[m]
  Repeat until SP[1..n] all filled
  Begin
    Select point p[j] with maximum distance to
      the adjacent points in SP (sp[1] and sp[n]
      initially)
    Add p[j] TO SP
  End
  Return SP
End
  
```

Figure 2. Pseudo code of the PIP identification process

The distance metric for distance measurement (i.e. calculation of fluctuation value of a data point) is the vertical distance (VD) between a test point p_3 and the line connecting the two adjacent PIPs, i.e.,

$$VD(p_3, p_c) = |y_c - y_3| = \left| \left(y_1 + (y_2 - y_1) \cdot \frac{x_c - x_1}{x_2 - x_1} \right) - y_3 \right| \quad (1)$$

where $x_c = x_3$. Eqn. (1) is intended to capture the fluctuation of the data sequence and the highly fluctuated points (or locally extreme points) would be considered as PIPs.

To illustrate the identification process, Fig.3 shows the step-by-step results. Here, the number of data points in the input sequence P and query sequence Q (or the required number of PIPs) are 10 and 5 respectively, i.e., $m=10$ and $n=5$.

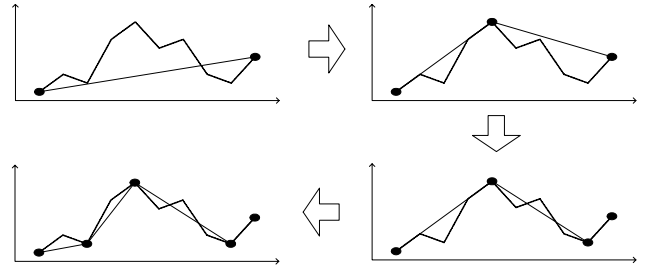


Figure 3. Identification of 5 perceptually important points

3.2 Building Data Point Importance List

Inspired from the above idea, a reordering mechanism for the data points in a time series is proposed in paper [32]. Instead of only identifying n PIPs from the longer sequence m for pattern matching, only one time series is provided in the current scenario. Therefore, all the data points in the time series P will go through the PIP identification process. In other words, the number of PIPs to be identified is equal to the number of data points in the given time series. However, the sequence of the data points being identified is the main focus.

The sequence of PIPs is ordered according to Fig.4. It is the same as the original process except the PIP identification process continues until all the data points in the given time series P have been processed and appended to the list L .

```

Function Build_list (P)
  Input: sequence P[1..m]
  Output: sequence L[1..m]
Begin
  Set l[1]=p[1], l[2]=p[m]
  Repeat until l[1..m] all filled
  Begin
    Select point p[j] with maximum distance to
    the adjacent points in the list (l[1] and
    l[2] initially)
    Append point p[j] TO L
  End
  Return L
End

```

Figure 4. Pseudo code of the data point importance list construction

According to Fig.4, the PIPs identified in the earlier iterations should be considered as more important than those points identified later. Therefore, they have higher importance. Continued with the example in the previous sub-section, Fig.5 shows the reordered sequence of the data points based on the PIP identification process. The points with smaller number label are more important (e.g. PIP 3 is more important than PIP 4). Table II shows the data point importance list built for this example. Besides the importance of the PIPs, their corresponding amplitude y and vertical distance measured are also stored in the table for future references. A point needed to be mentioned here is that the fluctuation value, i.e. the VD value, is not guaranteed to decrease with the decrease in importance. It is because the measurement of VD depends on the global shape captured in previous PIP identification iteration but not an overall shape. So, the change of such shape will affect the calculation of VD of the data points in the original time series compared with the pattern formed by the identified PIPs (e.g. PIPs with importance 6 and 7).

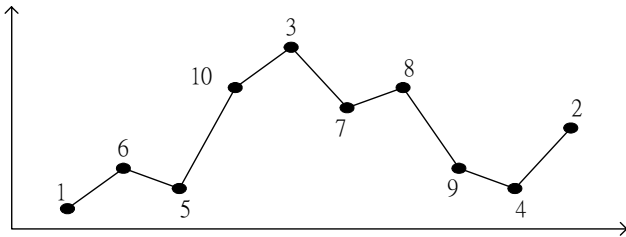


Figure 5. The order of points found by the PIP identification process

Table II. Example of data point importance list

Importance	x	y (Amplitude)	Fluctuation (VD)
1	1	0.1	n/a
2	10	0.5	n/a
3	5	0.9	0.62
4	9	0.3	0.28
5	3	0.3	0.20
6	2	0.4	0.20
7	6	0.6	0.15
8	7	0.7	0.20
9	8	0.4	0.10
10	4	0.7	0.10

With the construction of the data point importance list, the retrieval, analysis and processing of time series can be carried based on this list and different degrees of benefit, in terms of efficiency and effectiveness, can be obtained. A tree structure representation for storing the list is proposed in paper [32]. In this paper, the data point importance list is applied for dimensionality reduction to represent the time series in lower resolution for indexing.

4. Stock Time Series Indexing based on Clustering

To facilitate time series indexing, representing the time series in low dimension is of fundamental importance. It is because time series data with different lengths always coexists and measuring the distance among them is the necessary step of the indexing process. Furthermore, a mechanism for representing time series in a low resolution can improve the efficiency of the indexing algorithm and maintain the visualization ability. Reference [33] further shows that initializing the clusters' centroids on a low resolution of the data can improve the quality. Therefore, the overall process for the proposed stock time series indexing scheme can be classified as two main components: dimensionality reduction of the time series data to a lower resolution and the clustering process for time series indexing.

4.1 Dimensionality Reduction by Accessing the Data Point Important List

With the data point importance list L built for the time series P , the low resolution of the time series SP can be obtained by selecting n number of PIPs from the top of the list as Fig.6, where $n \ll m$. By applying this method, all the time series in the database can provide a lower resolution version of representation with uniform length for indexing and, therefore, dimensionality reduction can be achieved.

```

Function PIP_retrieval (L, n)
  Input:  sequence L[1..m]
  Output: sequence SP[1..n]
Begin
  For i = 1 to n
  Begin
    Append point l[i] TO SP
  End
  Return SP
End

```

Figure 6. Data point retrieval from the data point important list for dimensionality reduction

Referring to the example in section 3.2, if $n=3$, it means that the first 3 PIPs, (10, 0.5), (1, 0.1) and (5, 0.9), should be retrieved from the data point important list to represent the time series in lower resolution. The visualization result of the low resolution of the sample time series is as Fig.7.

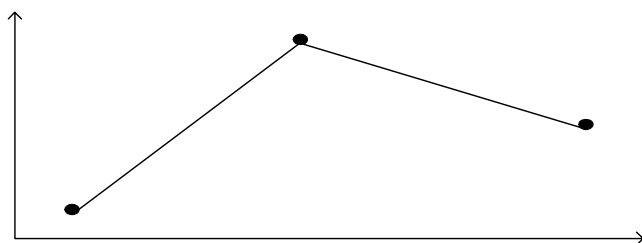


Figure 7. Low resolution ($n=3$) of the sample time series in Fig.5

4.2 Building Index by Clustering

After we transformed all the time series in the database to a lower resolution, we can group the time series with similar shape together for indexing. Clustering is a common method for finding structure in given data [9, 11], in particular for finding structure related to time. There are many popular clustering techniques developed, such as hierarchical, nearest neighbor, and k -means algorithms. In the data mining context, the most common one perhaps is k -means algorithm [34] and it is good enough for our time series indexing process.

In k -means clustering, each cluster is represented by the center of the cluster, centroid. The k -means algorithm is implemented in 4 steps: (1) cluster objects into k nonempty subsets, (2) compute seed points as the centroids of the clusters of the current partition, (3) Assign each object to the cluster with the nearest seed point and (4) go back to the step 2, stop when no more new assignment or excess the maximum number of iteration.

Suppose that there exist N objects, x_1, x_2, \dots, x_N , in the database, and we know that they fall into k compact clusters, $k \ll N$. Let M_i be the mean of the vectors, that is, the cluster centroid, in cluster i . If the clusters are well separated, a minimum distance classifier can be used to separate them. That is, we can say that x is in cluster i if $|x - M_i|$ is the

minimum of all the k clusters. This suggests the procedure in Fig.8 for finding the k means.

```

Function k_means_clustering (N)
  Input:  object x[1..N]
Begin
  Make initial guesses for the means  $M_1, M_2, \dots, M_k$ 
  Until there are no changes in any mean or
  excess the maximum number of iteration
  Use the estimated means to classify the
  objects into clusters
  For i from 1 to k
  Replace  $M_i$  with the mean of all the
  objects for cluster  $i$ 
  End for
  End Until
End

```

Figure 8. k -means clustering process

To index the time series database by clustering process, all the time series in the database are the input objects for the clustering process and the low resolution of these time series is considered as the input feature vectors. As the low resolution of the time series n is smaller or equal to the smallest length among all time series in the database, all the time series can be indexed even though the length of original time series in the database are different. Finally, each of the time series will group into one cluster among k clusters as Fig.9.

The distance between the low resolution of the time series (SP) from time series (P) and the cluster centroid M_i can be computed using direct point-to-point comparison, that is,

$$Dist(SP, M_i) = \sqrt{\frac{1}{n} \sum_{j=1}^n (sp_j - m_{j,k})^2} \quad (2)$$

for all centroids, M , during the clustering process.

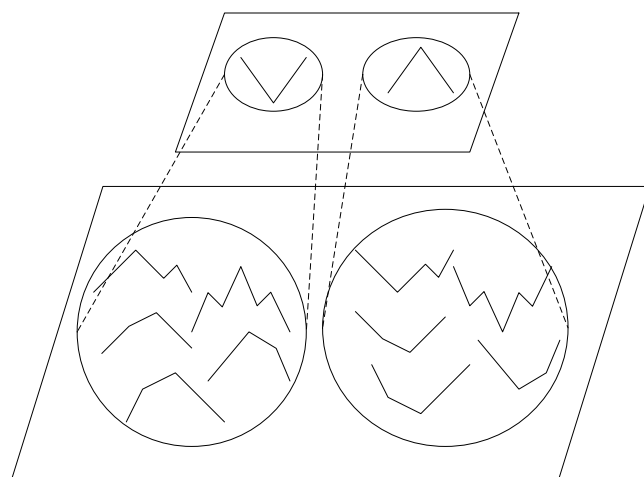


Figure 9. Time series clustering

4.3 Updating the Index

In addition, it is an easy task to update the index when adding a new time series P_{new} . The data point importance list of the new time series should first be built and the resolution of this time series can then be reduced to n , SP_{new} . Afterwards, it can be added to the cluster i that $|SP_{new}-M_i|$ is the minimum of all the k clusters, or else, a new cluster has to be created for this new time series if $|SP_{new}-M_i|>\lambda$, where λ is an user defined threshold (Fig.10).

```

Function Incremental_update (P_new)
  Input: sequence P_new[1..m]
Begin
  L_new = Build_List (P_new)
  SP_new = PIP_retrieval (L_new)

  Find min(|SP_new-M_i|) for all cluster (M_1 to M_k)

  If min(|SP_new-M_i|)>\lambda
    Create a new cluster M_{k+1} for P_new
  Else
    Append P_new to cluster i
  End If
End

```

Figure 10. Incremental updating of the index

4.4 Query the Index

After building the index, the users can query the time series database by first providing the shape of the time series, Q (i.e. query by example) with n number of data points or reducing the dimension of Q to n by building the proposed data point important list, where n is same as the low resolution of the time series in the database. Direct comparison can be applied between Q and the cluster centroids, M , using Eq.2. By identifying which cluster the query sequence belongs to (with minimum distance), only the time series which belongs to this cluster is needed to evaluate. By this method, the number of time series needed to examine can be reduced while the overall shape of the time series can still be preserved. Fig.11 shows the query procedure.

```

Function Query (Q)
  Input: sequence Q[1..n]
  Output: sequence P[i..m]
Begin
  Find min(|Q-M_i|) for all cluster (M_1 to M_k)
  Examine all the time series belongs to M_i and
  find P which with the minimum distance with Q
  Return P
End

```

Figure 11. Query process for our indexing approach

5. Simulation Results

In this section, the performance of the proposed time series indexing algorithm will be reported. The dataset used was taken from the opening price time series of the Hong Kong stock market. 200 stock time series with the length between 251 and 1685 were considered. And the average

length is 1537. Five different time series were randomly selected from the dataset to serve as the query series and the figures of all the experiments were the average of the five results.

Firstly, the effects on space consumption when increasing the resolution, which are the number of PIP retrieved n and the number of cluster k , were tested. Obviously, more space was required when increasing either the number of PIP or the number of cluster (Fig.12).

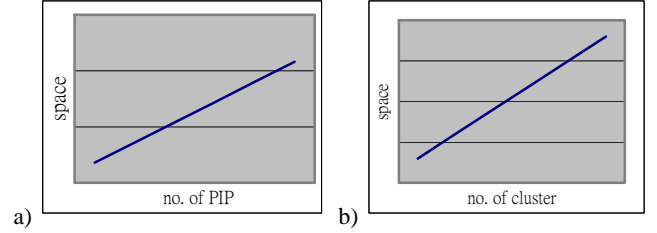


Figure 12. (a) Space consumption vs. number of PIP and (b) Space consumption vs. number of cluster

Secondly, the CPU cost (i.e. speed) and the number of iteration needed to converge for building the index with different parameters settings, which are the resolution (number of PIP retrieved, n) and the number of cluster k , were tested. The number of clustering is fixed to 4. As shown in Fig.13, the increase of number of PIP raised the time for building the index. It is because the increase of the number of PIP augments the dimension of the input feature vector for the clustering process. Moreover, it was more difficult for the clustering process to converge in a higher dimension. More iterations were also needed (Fig.14). When the number of PIP was set to 15 and 20, the maximum number of iteration (i.e. 500 cycles) was reached. It means that the index built by these two experiments might have false dismissals as the clusters were not well separated. Fig.15 shows the shapes of the cluster centroids after the clustering process with different numbers of PIP. Similar shapes were obtained in different resolutions but more detail of the shapes was shown in higher resolution.

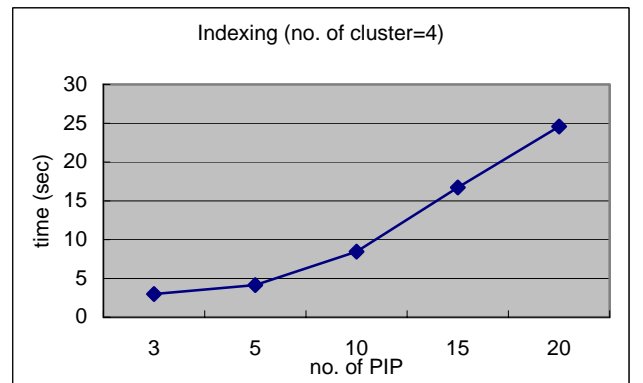


Figure 13. CPU cost for building the index against number of PIP (n)

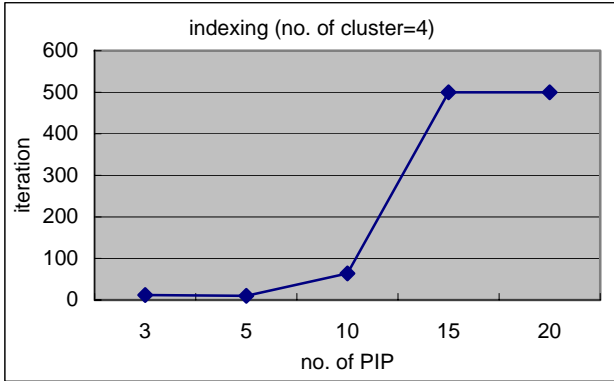


Figure 14. Number of iteration for building the index against number of PIP (n)

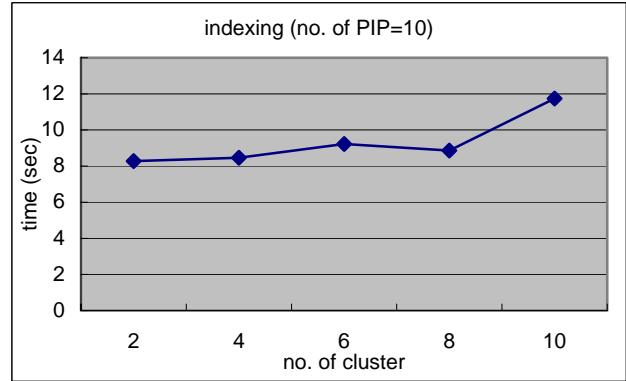


Figure 16. CPU cost for building the index against number of cluster (k)

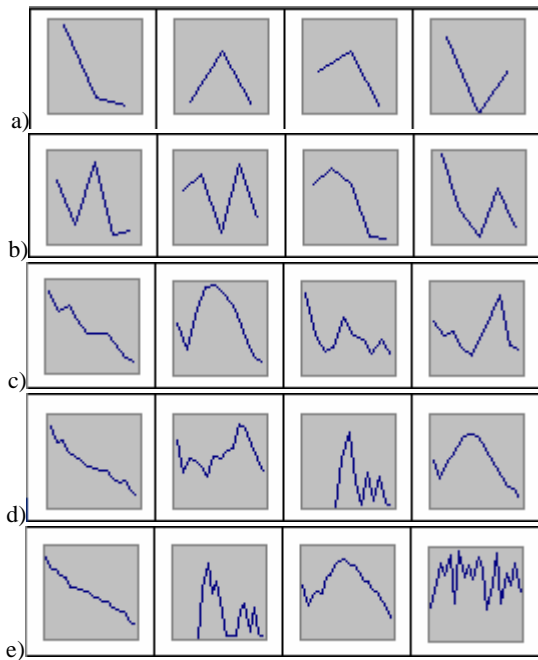


Figure 15. Visualization of clusters' centroids (a) $n=3$ (b) $n=5$ (c) $n=10$, (d) $n=15$ and (e) $n=20$

Then, the number of PIP was fixed to 10. When increasing the number of cluster, the time for building the index increased more stably compared to the increase of the number of PIP. The result of the experiment shows that the dimension of the input feature vector dominates the time for the indexing process when comparing to the number of clustering. Furthermore, Fig.17 shows that the dependency between the number of cluster and the number of iteration needed for building the index was low but it should be in direct proportion.

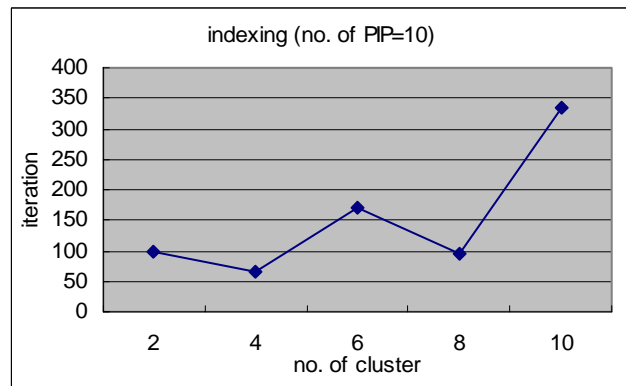


Figure 17. Number of iteration for building the index against number of cluster (k)

Thirdly, the pruning power and the CPU cost of the proposed algorithm were measured with different n and k during the query process. Pruning power is defined as Eq. (3) [14].

$$P = \frac{\text{Number of object that must be examined}}{\text{Number of object in the database}} \quad (3)$$

In this part, the effect of n was tested first. The number of cluster k was fixed to 4. As shown in Fig.18 and Fig.19, both the pruning power and the query time were independent to the number of PIP but rather dependent on the number of cluster. Then, n is fixed to 10, and the effect of using different number of cluster k was tested. As shown in Fig.20 and Fig.21, the increase of number of clustering improved the query performance. However, as the size of the dataset was not large, the improvement was kept in stable after 4 clusters were used.

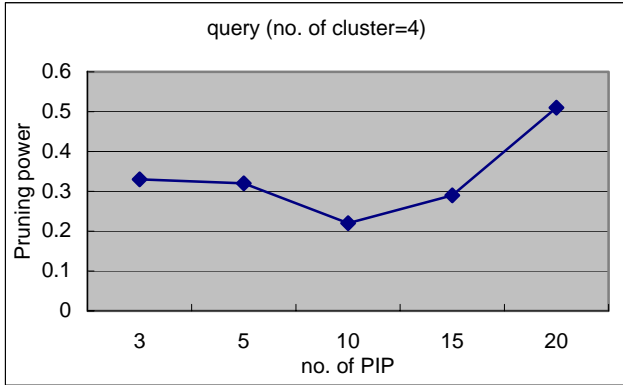


Figure 18. Pruning power against number of PIP (n)

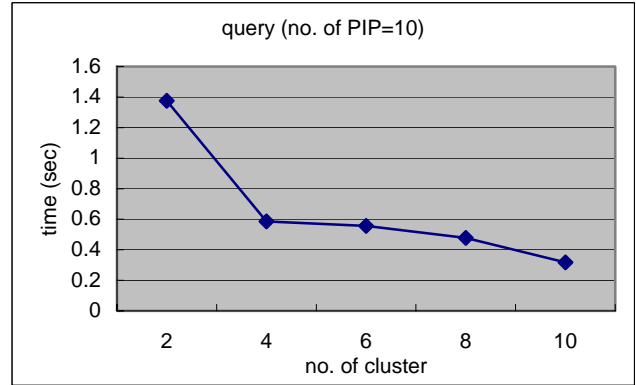


Figure 21. Query time against number of cluster (k)

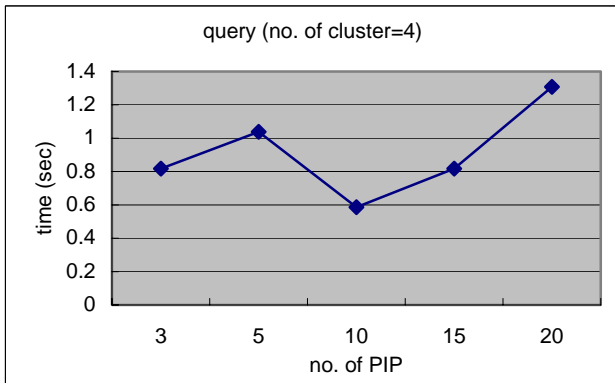


Figure 19. Query time against number of PIP (n)

In the above experiments, although time is needed to build the index of the time series database, this is only a one-time process. On the other hand, the query time can be reduced in an obvious degree with no false dismissals when the clustering process could be converged before reaching the maximum number of iteration. It can be easily achieved by considering the low resolution (the number of PIP n) of the input feature vector. Moreover, the time series database can be updated easily as suggested in section 4.3. Finally, Fig.22 shows the comparison on the query time between the sequential scanning of the time series database and the proposed indexing approach ($n=10$ and $k=4$). Obviously, the proposed approach outperformed the sequential scanning and the query time could be kept stable when increasing the number of object in the database.

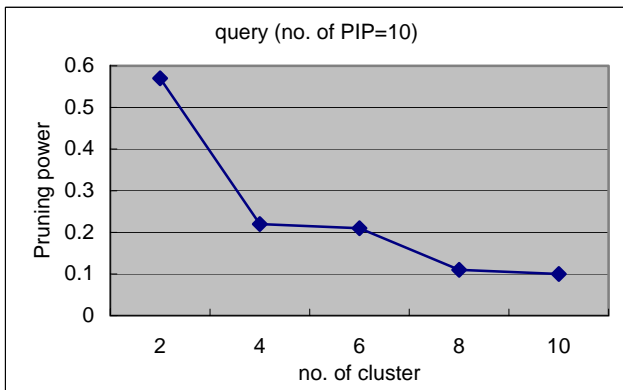


Figure 20. Pruning power against number of cluster (k)

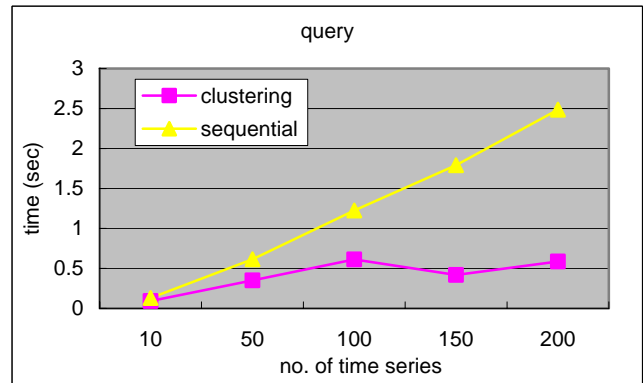


Figure 22. Comparison on the query time between sequential scanning and the proposed indexing approach

6. Conclusions

In this paper, a stock time series indexing approach is proposed. It is based on data point importance to reduce the dimension of the time series data. Then, the clustering process is applied to the low resolution of the time series for indexing. This method can solve the problem of the coexisting of time series with different lengths in the database. It is not only efficient but also effective. The proposed approach is customized for stock time series data

which has its unique behaviors. As demonstrated by the experiments, the proposed approach speeds up the time series query process while no false dismissals can be guaranteed. In addition, the proposed approach can handle the updating problem of the time series database without any difficulty. For further development, it is possible to further enhance the indexing process by applying the concept of hierarchical indexing. That is, multiple-resolution clustering technique can be applied and clustered the time series data from low resolution to high resolution.

References

1. G. Das, D. Gunopulos and H. Mannila, "Finding Similar Time Series," *In proceedings of the 1st European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp.88-100, 1997.
2. R. Agrawal, C. Faloutsos and A.N. Swami, "Efficient similarity search in sequence databases," *In proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, pp.69-84, 1993.
3. Faloutsos, M. Ranganathan and Y. Manolopoulos, "Fast subsequence matching in time series databases," *In proceeding of the ACM SIGMOD International Conference on Management of Data, Minneapolis*, pp. 419-429, 1994.
4. K. Chu and M. Wong, "Fast time-series searching with scaling and shifting," *In proceedings of the 18th ACM Symposium on Principles of Database Systems*, pp. 237-248, 1999.
5. Refiei, "On similarity-based queries for time series data," *In proceedings of the 15th International Conference on Data Engineering*, pp. 410-417, 1999.
6. D. Rafiei and A.O. Mendelzon, "Querying Time Series Data Based on Similarity", *IEEE Transactions on Knowledge and Data Engineering*, 12 (5), pp.75-693, 2000.
7. T. Kahveci and A. Singh, "Variable length queries for time series data," *In proceedings of the 17th International Conference on Data Engineering*, pp. 273-282, 2001.
8. Popivanov and R. J. Miller, "Similarity search over time series data using wavelets," *In proceedings of the 18th International Conference on Data Engineering*, pp. 212-221, 2002.
9. C. Wang and X. S. Wang, "Supporting content-based searches on time series via approximation," *In proceedings of the 12th International Conference on Scientific and Statistical Database Management*, pp. 69-81, 2000.
10. Y. Wu, D. Agrawal and A. El Abbadi, "A comparison of DFT and DWT based similarity search in time-series databases," *In proceedings of the 9th ACM CIKM International Conference on Information and Knowledge Management*, pp. 488-495, 2000.
11. K.V. Kanth, D. Agrawal and A. Singh, "Dimensionality reduction for similarity searching in dynamic databases," *In proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 166-176, 1998.
12. Ferhatosmanoglu, E. Tuncel, D. Agrawal and A. El Abbadi, "Approximate nearest neighbor searching in multimedia databases," *In proceedings of the 17th IEEE International Conference on Data Engineering*, pp. 503-511, 2001.
13. B. Yi and C. Faloutsos, "Fast time sequence indexing for arbitrary Lp norms," *In proceedings of the 26th International Conference on Very Large Databases*, pp. 385-394, 2000.
14. E. Keogh, K. Chakrabarti, M. Pazzani and S. Mehrotra, "Locally adaptive dimensionality reduction for indexing large time series databases," *In proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 151-162, 2001.
15. Y. W. Huang and P. S. Yu, "Adaptive Query Processing for Time Series Data", *In proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp 282-286, 1999.
16. B. Yi, H.V. Jagadish and C. Faloutsos, "Efficient Retrieval of Similar Time Sequences Under Time Warping," *In proceedings of the 14th International Conference on Data Engineering*, pp.201-208, 1998.
17. E. Keogh, "Exact Indexing of Dynamic Time Warping," *In proceedings of the 28th International Conference on Very Large Data Bases Conference*, pp.406-417, 2002.
18. E. Keogh and S. Kasetty, S, "On the Need for Time Series Data Mining Benchmarks: A Survey and Empirical Demonstration", *In proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp 102-111, 2002.
19. J. Lin, E. Keogh, S. Lonardi and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," *In proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2003.
20. E. Keogh and M. Pazzani, "An Indexing Scheme for Fast Similarity Search in Large Time Series Databases," *In proceedings of the 11th International Conference on Scientific and Statistical Database Management*, pp.56-67, 1999.
21. H.A. Henrik and D. Z. Badal, "Using Signature Files for Querying Time Series Data", *In proceedings of Principles of Data Mining and Knowledge Discovery*, pp 211-220, 1997.
22. T. Sellis, N. Roussopoulos, C. Faloutsos, "The R+-Tree: A Dynamic Index For Multi-Dimensional Objects", *The Very Large Database Journal*, 1987.
23. I. Zwir and E.H. Enrique, "Qualitative Object Description: Initial Reports of the Exploration of the Frontier," *In proceedings of the Joint EUROFUSE-SIC'99 International Conference*, 1999.
24. K. Bennett, U. Fayyad and D. Geiger, "Density-based indexing for approximate nearest-neighbor queries," *In proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*, pp. 233-243, 1999.
25. X. Ge and P. Smyth, "Deformable Markov model templates for time-series pattern matching," *In proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.81-90, 2000.
26. F.L. Chung, T.C. Fu, R. Luk and V. Ng, "Flexible time series pattern matching based on perceptually important points," *International Joint Conference on Artificial Intelligence Workshop on Learning from Temporal and Spatial Data*, pp.1-7, 2001.
27. D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *The Canadian Cartographer*, vol.10, no.2, pp.112-122, 1973.
28. J. Hersherberger and J. Snoeyink, "Speeding up the Douglas-Peucker line-simplification algorithm," *In the proceedings of the 5th Symposium on Data Handling*, pp.134-143, 1992.

29. C.S. Perng, H. Wang, R. Zhang and D. Parker, "Landmarks: A new model for similarity-based pattern querying in time series databases," *In proceedings of the 16th International Conference on Data Engineering*, pp.33-42, 2000.
30. B. Pratt and E. Fink, "Search for patterns in compressed time series," *Image and Graphics*, vol.2, no.1, pp.89-106, 2002.
31. E. Fink and B. Pratt, "Indexing of compressed time series," *Data Mining in Time Series Databases*, pp.51-78, 2003.
32. T.C. Fu, F.L. Chung, R. Luk and C.M. Ng, "A Specialized Binary Tree for Financial Time Series Representation," *The 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining Workshop on Temporal Data Mining*, 2004.
33. C. Ding, X. F. He, H. Y. Zha and H. D. Simon, "Adaptive dimension reduction for clustering high dimensional data," *In proceedings of the 2nd IEEE International Conference on Data Mining*, pp. 147-154, 2002.
34. U. Fayyad, C. Reina and P. Bradley, "Initialization of Iterative Refinement Clustering Algorithms," *In proceedings of the 4th International Conference on Knowledge Discovery and Data*, pp. 194-198, 1998.