A New Dynamic Indexing Structure for Searching Time-Series Patterns

Zi-Yu Lin[1], Yong-Sheng Xue[1], and Xiao-Hua Lv[2]

[1] Department of Computer Science of Xiamen University
361005 Xiamen, China
cainiu@263.net, ysxue@jingxian.xmu.edu.cn
[2] College of Information and Electron of Zhejiang University of Sciences
310018 Hangzhou, China
bowen95@sohu.com

Abstract. We target at the growing topic of representing and searching time-series data. A new MABI (Moving Average Based Indexing) technique is proposed to improve the performance of the similarity searching in large time-series databases. Notions of Moving average and Euclidean distances are introduced to represent the time-series data and to measure the distance between two series. Based on the distance reducing rate relation theorem, the MABI technique has the ability to prune the unqualified sequences out quickly in similarity searches and to restrict the search to a much smaller range, compare to the data in question. Finally the paper reports some results of the experiment on a stock price data set, and shows the good performance of MABI method.

1 Introduction

Time-series data are of growing importance in the field of database application, more and more researchers are devoting themselves to developing powerful and useful data mining tools based on time-series database. Nowadays, we can see the application of these tools in many fields such as medicine, finance, meteorology, etc. In these application fields, large amounts of data are constantly added to systems, for example, MACHO project, an astronomy database, contains more than 0.5 TB data and receives a large amount of new data everyday [2].For this reason, there must be effective methods for solving problems such as representation and indexing of time-series.

Here we propose a new index mechanism—MABI (Moving Average Based Indexing), which is not only easy to implement but also has desirable performance. It has great capability in pruning those unqualified time-series and can also be applied in other fields

The remainder of this paper is organized as follows: Section 2 discusses relative work in this field. Section 3 gives some preliminary knowledge for our method. Section 4 presents MABI method in detail, which involves some relating definitions, useful theorems and algorithms. Section 5 gives performance analysis. Section 6 presents our conclusions.

S. Wang et al. (Eds.): ER Workshops 2004, LNCS 3289, pp. 290–301, 2004.

2 Related Work

R-tree is a well-known and wildly-used indexing mechanism. At the same time, some new indexing methods with good performance are also proposed in recent years. In [9], the authors proposed EMDF (Extended Multidimensional Dynamic index File) as indexing structure for time-series patterns. EMDF is a multidimensional spatial file to store and retrieve time-series patterns represented as multi-key records and is used for preselection process which select out a small set of candidate frames from the entire set, through this way, speedup in the pattern matching process is achieved. However, EMDF exists some disadvantages, for example, it has to maintain pointers for dynamic range tables which is very large in this method. In order to avoid these defection and enhance the performance, a new improved dynamic indexing structure for efficient handling of time-series was presented in [10]—TIP-index (TIme-series Patterns index). As far as structure is concerned, these two methods are similar. A TIP-index consists of frame identifier blocks, a dynamic region table and hierarchical directory blocks. However unlike EMDF, TIP-index only has s single dynamic region table which represents all dimensions. In the aspects of inserting operation and search, TIP-index can be more efficient than EMDF, furthermore, the former can achieve balanced performance for different types of data with special value distribution, however, the latter performs poorly when encountering such circumstances. In [11], the authors gave a novel method called STB-index (Shape-To-Vector-index), in which, index is formed by creating bins that contain time series subsequences of approximately the same length, for each bin, they can quickly calculate a lower-bound on the distance between a given query and the most similar element of the bin, this bound allows them to search the bins in best first order and to prune some bins from search space without having to examining the contents. Some other methods include: vantage-point-tree index in [12], using maximum and minimum to create index in [13], multilevel hierarchical structure based on feature of time series in [14], etc. There are also some well-known indexing mechanism such as KD-trees[4],K-D-B-trees[5] and Quad-trees[6], however, they cannot be applied in the field of time-series, because time-series databases are different from traditional databases in that the former must keep not only the value but also the time when the value is produced.

3 Preliminary

3.1 Representation of Data

For clarity we will refer to ‘raw’, unprocessed temporal data as time-series, and a piecewise representation of a time-series as a sequence. At the same time, a “series” will denote a time-series or a sequence. An effective representation of data is the prerequisite for constructing a dynamic indexing mechanism with desirable performance. Here we introduce a well-known and widely-used representation method in time-series field.

PAA (Piecewise Aggregate Approximation) is a method easy to understand [3]. A time series C of length n can be represented in N dimensional space by a vector $\overline{C} = \overline{c}_1, \overline{c}_2, .., \overline{c}_N$, and we assume that N is a factor of n. The i[th] element of $\overline{C}$ is calculated as follows:

$$\overline{c}_i = \frac{N}{n} \sum_{\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} c_j \tag{1}$$

This simple technique is surprisingly competitive with the more sophisticated transforms, so we take PAA as our method of data representation.

3.2 Moving Average

Moving average data are widely used in stock data analysis. Their primary use is to smooth out short term fluctuations and depict the underlying trend of a stock. The computation is simple, now we give the definition:

Definition 1. For a series $\vec{x} = [x_t]$ (t=0,1,…,n-1), which contains n data points, its M-l moving average is computed as follows: the mean is computed for an l-point-wide window placed over the end of the series; this will give the moving average for point $n - \lfloor l/2 \rfloor$; the subsequent values are obtained by stepping the window through the beginning of the series, one point at a time. This will produce a moving average of length n-l+1; We can use $M\text{-}l(\vec{x})$ to represent the M-l moving average of $\vec{x}$.

3.3 Euclidean Distance

In the field of time series, Euclidean distance is a method used to measure the distance between two series. For two series, $\vec{x} = \{x_0, x_1, \ldots, x_{n-1}\}$, $\vec{y} = \{y_0, y_1, \ldots, y_{n-1}\}$, the Euclidean distance between them is computed as follows:

$$\boldsymbol{D}(\vec{x}, \vec{y}) = \left(\sum_{i=0}^{n-1} (x_i - y_i)^2 \right)^{\frac{1}{2}} \tag{2}$$

We usually use Euclidean distance to determine whether two series are similar or not. If $\boldsymbol{D}(\vec{x}, \vec{y}) < \varepsilon$ (ε is a threshold set by users), then we may say that the two series are similar.

3.4 Two Novel Theorems Based on Moving Average

Now we give two very important theorems.

Theorem 1. For two sequences $\vec{x}=\{x_0, x_1, \ldots, x_{n-1}\}$ and $\vec{y}=\{y_0, y_1, \ldots, y_{n-1}\}$, where $1 \le l \le n$, there *M-l* moving average are *M-l* ($\vec{x}$) and *M-l* ($\vec{y}$) respectively, then they must satisfy the following formula:

$$\boldsymbol{D}(M\text{-}l(\vec{x}), M\text{-}l(\vec{y})) \le \boldsymbol{D}(\vec{x}, \vec{y})$$

The conditions for $\boldsymbol{D}(M\text{-}l(\vec{x}), M\text{-}l(\vec{y})) = \boldsymbol{D}(\vec{x}, \vec{y})$ are as follows:

(1) l=1;

(2) l>1 and $\vec{x}$ is identical to $\vec{y}$;

In other circumstances, $\boldsymbol{D}(M\text{-}l(\vec{x}), M\text{-}l(\vec{y})) < \boldsymbol{D}(\vec{x}, \vec{y})$.

Here we call Theorem 1 "distance reducing theorem". Due to the restriction of the pages, we would not give the proof here.

According to Theorem 1, we can get another useful theorem.

Theorem 2. Under the same similarity criterion, if two sequences are similar, then their *M-l* moving average are also similar.

Proof: It is easy to prove. Let $\vec{x}=\{x_0, x_1, \ldots, x_{n-1}\}$ and $\vec{y}=\{y_0, y_1, \ldots, y_{n-1}\}$, where $1 \le l \le n$, if they are similar, then there exists $\boldsymbol{D}(\vec{x}, \vec{y}) < \varepsilon$, according to Theorem 1, $\boldsymbol{D}(M\text{-}l(\vec{x}), M\text{-}l(\vec{y})) \le \boldsymbol{D}(\vec{x}, \vec{y})$, then $\boldsymbol{D}(M\text{-}l(\vec{x}), M\text{-}l(\vec{y})) < \varepsilon$, so *M-l* ($\vec{x}$) and *M-l* ($\vec{y}$) are similar. ■

Definition 2. For a sequence *s*, its *M_l moving average is M_l (s),* Euclidean distance between *s* and horizontal axis is represented as *Ds*, Euclidean distance between *M_l (s)* and horizontal axis is D_M, DRR (Distance Reducing Rate) *r* between *s* and *M_l (s)* is defined as: $r = (Ds - D_M)/Ds$.

Here suppose there are two sequences s_1 and s_2, their *M_l moving average are M_l* (s_1) and *M_l* (s_2), Euclidean distances between the four sequences and horizontal axis are Ds_1 , Ds_2, D_{M1} and D_{M2} respectively. Let r_1 denotes DRR (Distance Reducing Rate) between s_1 and *M_l* (s_1), r_2 denotes DRR between s_2 and *M_l* (s_2). We now give the most important theorem, it is the foundation of MABI method.

Theorem 3. For two sequences s_1 and s_2, if $Ds_1>\varepsilon$, then they must satisfy the equation: $\frac{Ds_1 \times r_1 - 2\varepsilon}{Ds_1 + \varepsilon} < r_2 < \frac{Ds_1 \times r_1 + 2\varepsilon}{Ds_1 - \varepsilon}$

Proof: Suppose the distance between s_1 and s_2 is less than ε, namely, $D(s_1,s_2) < \varepsilon$, according to triangular inequality theorem [15], we can get that $Ds_1-Ds_2 < D(s_1,s_2)$ and $Ds_2-Ds_1 < D(s_1,s_2)$, namely, $|Ds_1-Ds_2| < D(s_1,s_2)$, so we can get $|Ds_1-Ds_2| < \varepsilon$; According to Deduction 1, $M\_l(s_1)$ and $M\_l(s_2)$ are also similar, so we get that $|D_{M1}-D_{M2}| < \varepsilon$, then $|Ds_1-Ds_2|+|D_{M1}-D_{M2}| < 2\varepsilon$, according to the knowledge about absolute value we know that $|a|-|b| \le |a-b| \le |a|+|b|$, so we get that $|(Ds_1-Ds_2)-(D_{M1}-D_{M2})| \le |Ds_1-Ds_2|+|D_{M1}-D_{M2}|$, so, $|(Ds_1-Ds_2)-(D_{M1}-D_{M2})| < 2\varepsilon$, then, $|(Ds_1-D_{M1})-(Ds_2-D_{M2})| < 2\varepsilon$, namely, $|Ds_1 \times r_1 - Ds_2 \times r_2| < 2\varepsilon$, because $Ds_2 = Ds_1 \pm \varepsilon$, then we have $|Ds_1 \times r_1 - (Ds_1 \pm \varepsilon) \times r_2| < 2\varepsilon$, when $Ds_1 \le \varepsilon$, $|Ds_1 \times r_1 - (Ds_1 \pm \varepsilon) \times r_2| < 2\varepsilon$ must be true for all the values of distance reducing rate, when $Ds_1>\varepsilon$, then we get: $\frac{Ds_1 \times r_1 - 2\varepsilon}{Ds_1 \pm \varepsilon} < r_2 < \frac{Ds_1 \times r_1 + 2\varepsilon}{Ds_1 \pm \varepsilon}$,so we can get: $\frac{Ds_1 \times r_1 - 2\varepsilon}{Ds_1 + \varepsilon} < r_2 < \frac{Ds_1 \times r_1 + 2\varepsilon}{Ds_1 - \varepsilon}$. ■

Here we call Theorem 3 "DRR relation theorem", with which we can prune quickly those unqualified sequences when searching for similar sequences of a given query sequence, because for a query *q,* if a sequence *s* is similar to *q*, it is a prerequisite that *s* and *q* satisfy DRR relation theorem.

4 MABI Method

MABI method makes full use of DRR (Distance Reducing Rate) relation theorem presented above and can prune those unqualified sequences quickly, through this way, the range of search is confined to very small area, which leads to desirable performance enhancement.

4.1 Building MABI Index Tree

In MABI index tree, the nodes are classified into leaf nodes and non-leaf nodes, moreover, non-leaf nodes include a root node and many mid nodes. The structures of leaf nodes and non-leaf nodes are different. The structure of non-leaf node is as Figure 4.1, in which, *range_node* records the DRR range that the node represents, *value* records the information of DRR range, *node_pointer* points to its son node whose *range_node* is equal to *value*. For a leaf node, it contains only two items: *range_node* and *table_pointer*, *table_pointer* points to information table *T*. *T* includes the follow-

ing information: *r*, *address* and *euclidean*, where *r* is a DRR, *address* denotes where a sequence is stored, *euclidean* denotes the Euclidean distance between a sequence and horizontal axis. The maximum number of records that *T* can contain is *p*, and these records are arrayed in order according to *r*. If *r* of a sequence is covered by the *range_node* of a leaf node, then the information of the sequence will be recorded in *T* of the leaf node. When the number of records that a leaf node *t* contains reaches *p*, the leaf node *t* will automatically split its DRR range into *k* son DRR ranges of equal length and create *k* corresponding leaf nodes for these *k* son DRR ranges. Then the node *t* will put all the information of itself into the *k* new leaf nodes, after this, node *t* will change itself into a non-leaf node and record the information about its son nodes.

range_node	
range_node[1]	*value*
	node_pointer
range_node[2]	*value*
	node_pointer
……	……
	……
range_node[11]	*value*
	node_pointer

Fig. 4.1. The structure of non-leaf node of MABI index tree.

In order to build index tree from a time-series, we should first transform the time-series into a sequence with PAA algorithm, then we design a *b*-point-wide window placed at the beginning of the sequence, let the window move ahead along the sequence one point at a time, through this way we can get many subsequences of equal length, which will be used to build the index tree. The following algorithm in Figure 4.2 is used to insert a new subsequence *s* into a index tree.

4.2 Search Similar Sequences in MABI Index Tree

Here we presume that query *q* is a sequence got from a time-series with PAA algorithm. The search process involves three steps. In the first step, We will first calculate r_{q1}, which is a DRR between *q* and *M_l* (*q*), then compute *range*, which is a DRR range, $range = (\frac{Dq \times r_{q1} - 2\varepsilon}{Dq + \varepsilon}, \frac{Dq \times r_{q1} + 2\varepsilon}{Dq - \varepsilon})$, after this, we can find all leaf nodes whose *range_node* is covered by *range* (here if upper limit or lower limit of *range_node* is covered by *range*, then we call *range_node* is covered by *range*), and then select from *T* of these nodes those records whose r is covered by *range* and put them into a set *S*. In the second step, we use the condition $| Ds_1 - Ds_2 | < \varepsilon$ to continue pruning those unqualified elements from *S*. In the last step, we use condition $\boldsymbol{D}(s_1, s_2) < \varepsilon$ to get the final results. The algorithm is Figure 4.3.

```
Algorithm InsertSubsequence (s, root)
Input: s; root;
begin
    pointer:=root;//locate the pointer to root
    compute Ds and r of s;
  move along the tree according to r and reach leaf node u whose
 range_node covers r;
 add r to table T of node u;
 if the number of DRR that T contains reaches p then
    split the range_node of u into k parts of equal length;
  create new k leaf nodes for the k parts;
     move the information in T of u into the corresponding T of
   the k leaf nodes;
     recreate node u as a non-leaf node and record in it the in-
   formation of its k son nodes;
  endif;
end;
```

Fig. 4.2. Algorithm to insert a subsequence into MABI index tree.

Among the three steps, we use DRR relation theorem to prune many unqualified sequences in the first step, and use triangular inequality theorem [15] to do pruning job in the second step. For two sequences s_1 and s_2, according to triangular inequality theorem, we have $| Ds_1 - Ds_2 | < D(s_1, s_2)$, so if $\varepsilon < | Ds_1 - Ds_2 |$, then we surely have $\varepsilon < D(s_1, s_2)$, and s_1 and s_2 have no probability to be similar. The reason we design the second step is that the condition of $| Ds_1 - Ds_2 | < \varepsilon$ is more strict than the condition of $| Ds_1 - Ds_2 | + | D_{M1} - D_{M2} | < 2\varepsilon$ (which is the condition we use in the second step). Because D_{M1} and D_{M2} are the moving average of Ds_1 and Ds_2 respectively, so we have $| D_{M1} - D_{M2} | < \varepsilon_0$, where ε_0 is smaller than ε, so we say that $| Ds_1 - Ds_2 | < \varepsilon$ is more strict than $| Ds_1 - Ds_2 | + | D_{M1} - D_{M2} | < 2\varepsilon$. But we have to point out that the first pruning process is the critical aspect in enhancing query performance.

5 Performance Analysis

We do many experiments to get desirable results to support our new theory. Our experiments are executed on PC with one 1.3GHz CPU and 256M RAM running WIN XP. We download from internet the data of all the stocks in American stock market from 1980 to 2003 (*http://www.macd.cn*), which contains 1,228,764 records, then we get 10 time-series, each one contains 100,000 data points. We first transform the 10 time-series into 10 sequences of 10,000-point length with PAA algorithm. Here we let w=200, b=200, k=5, p=500, where w is the length of query q, b is the width of the window placed on the sequences, k and p are defined in section 4.1. Finally we get 9801 DRRs from every sequence. In order to show the pruning capability of DRR relation theorem, we select one sequence as an example. For this

```
Algorithm  Search(q,root, ε);
Input: q; root; ε
Output: sequences similar to q
begin
  //the beginning of the first step
  initialize a new queue Q;
  put root into Q;
  calculate M_2 (q), D_q ,range and rq_1;
  get an element from Q;
  pointer:= the element got from Q;
  while the node pointer points to is not a leaf node do
    for each pointer.range_node[i] do
      if range_node[i].value is covered by range then
        put range_node[i].node_pointer into Q;
      endif;
    endfor;
    get an element from Q;
    pointer:= the element got from Q;
  endwhile;
  while Q is not null do
    if pointer.range_node ⊆ range then
       put into S all records in T;
        //T belongs to the node that pointer
        //table_pointer points to;
    else // pointer.range_node are not completely
       //covered by range
       put into S those records in T whose r is covered by range;
         // T belongs to the node that pointer.
         //table_pointer points to;
    endif;
    get an element from Q;
    pointer:= the element got from Q;
  endwhile;
  //the end of the first step
  //the beginning of the second step
  for each s∈ S do
    if |Dq-s.Euclidean| > ε then delete s from S;
    endif;
  endfor;
  //the end of the second step
  //the beginning of the third step
  for each s∈ S do
  //let ls is the sequence that s.address points to
     if  the Euclidean distance between ls and q is more than ε
    then delete s from S;
    endif;
  endfor;
   //the end of the third step, get the result
  end;
```

Fig. 4.3. Algorithm to search similar sequences.

chosen sequence, its DRRs are distributed in the range of 0.9×10^{-2}~1.8×10^{-2} (as Figure 5.1 shows), then we get a subsequence of 200-point length as query sequence (as Figure 5.2 shows) and calculate its DRR, the result is r=1.628×10^{-2}, also we calculate the Euclidean distance between q and horizontal axis, the result is Dq=101.79, then we let $\varepsilon = Dq/f$ and endow f with different values to see how the performance of DRR relation theorem in pruning unqualified sequences changes. The experiment results are shown in Figure 5.3.

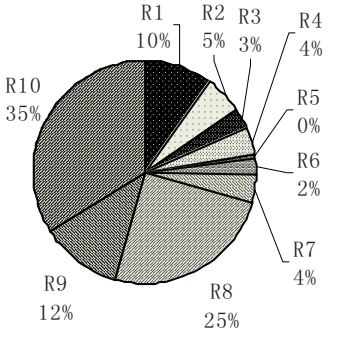

Fig. 5.1. Distribution of DRRs of the chosen sequence. The ranges are as follows: R1: $(0\sim1.0)\times10^{-2}$, R2: $(1.0\sim1.1)\times10^{-2}$, R3: $(1.1\sim1.2)\times10^{-2}$, R4: $(1.2\sim1.3)\times10^{-2}$, R5: $(1.3\sim1.4)\times10^{-2}$, R6: $(1.4\sim1.5)\times10^{-2}$, R7: $(1.5\sim1.6)\times10^{-2}$, R8: $(1.6\sim1.7)\times10^{-2}$, R9: $(1.7\sim1.8)\times10^{-2}$, R10: $(1.8\sim100)\times10^{-2}$.

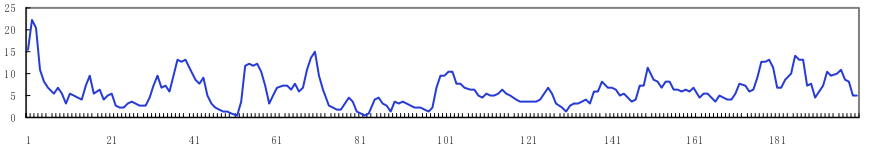

Fig. 5.2. The chosen query q of 200-point length.

We give in Figure 5.5 the pruning rate curve formed by changing the value of f. From Figure 5.5, we can draw the conclusion that, the larger the value of f is, the stronger the pruning capability is. At the same time, with the "expanding" of f, the extent of increase in pruning capability decreases, namely, when f is relative small, we can gain a great increase in pruning capability if we increase f a little. But when f reaches to certain value (f = 3000, for example), we can only achieve relative little enhancement from the increase of f.

In order to show the performance of MABI, we make it compete with sequential scanning optimized with Branch and Bound evaluation. To enable the results more reasonable and persuasive, we do 10 tests. In each test, we build index tree from one of the 10 sequences, and select randomly 9 200-point-long query sequences from the other 9 sequences, then we do search work with the index tree and the 9 query sequences. Finally we adopt the average of these results as our final results, which is as Figure 5.4.

f	1	500	1,000	2,000
pruning number	0	3551	5488	6538
pruning rate	0%	36.2%	56.0%	66.7%
range ($\times 10^{-2}$)	-99.19~ 100.81	1.226~ 2.024	1.427~ 1.826	1.527~ 1.727

f	5,000	10,000	20,000	100,000
pruning number	8791	9333	9521	9728
pruning rate	89.7%	95.2%	97.1%	99.3%
range ($\times 10^{-2}$)	1.588~1.668	1.608~1.648	1.618~1.638	1.626~1.630

Fig. 5.3. Performance of DRR relation theorem when *f* has different values.

f	length of sequence	sequential scanning (second)	MABI (second)	indexing speedup
10	100,000	980.86	287.94	5.41
500	100,000	737.84	179.96	4.10
2,000	100,000	727.67	86.38	8.42
10,000	100,000	725.19	28.43	25.51

Fig. 5.4. Performance contrast between MABI and Sequential scanning.

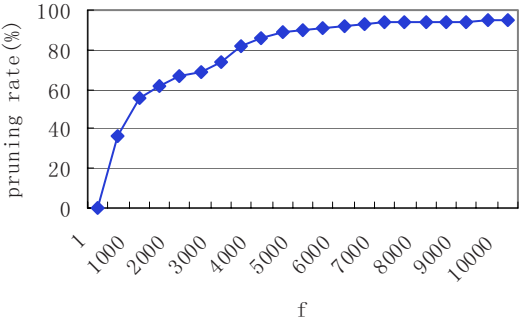

Fig. 5.5. Pruning rate curve.

Now we discuss a problem that whether MABI is an effective index mechanism. As presented in [8], there are several highly desirable properties for any indexing scheme:

- It should be much faster than sequential scanning.
- The method should require little space overhead.
- The method should be able to handle queries of various lengths.

- The method should be dynamic, which is to say it should allow insertions and deletions without requiring the index to be rebuilt.
- It should be correct, i.e. there should be no false dismissals (although probabilistic matching may be acceptable in some domains).

Our index mechanism has all the above properties except the third one. For the moment, we take sequential scanning as our method to deal with query shorter than *w*, and take the *long_query_searh* algorithm proposed in [11] to process query longer than *w*. In the future, we will continue our work to gain a better method to deal with this problem.

6 Conclusion

In this paper, we propose two novel theorems: distance reducing theorem and DRR relation theorem, based on which we propose MABI index mechanism, which is easy to implement and with little space overhead and desirable performance. However there exists a problem in our method, MABI can not effectively process queries of different length for the moment. We will do more work to perfect our method, at the same time, we will apply the two novel theorems proposed here in other fields.

References

1. Welch. D. & Quinn. P (1999) http://wwwmacho.mcmaster.ca/Project/Overview/status.html
2. Chan, K. & Fu, W. (1999). Efficient time series matching by wavelets. Proceedings of the 15th IEEE International Conference on Data Engineering.
3. Keogh, E,. Chakrabarti, K,. Pazzani, M. & Mehrotra (2000) Dimensionality reduction for fast similarity search in large time series databases. Journal of Knowledge and Information Systems.
4. Bently, J.L.(1975). Multidimensional binary search trees used for associative searching. Comm. ACM, Vol.18, No.9, 1975.
5. Robinson, J.T.(1981). The K-D-B-tree : A search structure for large multidimensional dynamic indexes. Proc. of Intl. Conf. on Management of Data, ACM SIGMOD, 1981.
6. Finkel, R. A. and Bently, J.L.(1974). Quad Trees: A data structure for retrieval on composite keys. Acta Informatica 4,1974.
7. Agrawal, R., Faloutsos, C., & Swami, A.(1993). Efficient similarity search in sequence databases. Proceedings of the 4th Conference on Foundations of Data Organization and Algorithms.
8. Faloutsos, C., & Lin, K. (1995). Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In Proc. ACM SIGMOD Conf., pp 163-174.
9. Kim, Y. I., Ryu, K. H. and Park, Y.(1994). Algorithms of improved multidimensional dynamic index for the time-series pattern. Proc. of the 1st KIPS Spring Conf., Vol. 1, No. 1, 1994.
10. Kim, Y.I., Park, Y., Chun, J.(1996).A dynamic indexing structure for searching time-series patterns. Proceedings of 20th International Computer Software and Applications Conference, 21-23 Aug., 1996. Pages:270–275.

11. Keogh, E.J., Pazzani, M.J.(1999). An indexing scheme for fast similarity search in large time series databases. Proceedings of the 11th International Conference on Scientific and Statistical Database Management, 28-30 Jul, 1999. Pages:56-57.
12. Bozkaya, T. and Ozsoyoglu, M. Indexing large metric spaces for similarity search queries. ACM Transactions on Database Systems, Volume 24, Issue 3, September, 1999. Page: 361 - 404.
13. Park, S., Kim, S. and Chu, W.W.(2001).Segment-based approach for subsequence searches in sequence databases. Proceedings of the 2001 ACM symposium on Applied computing, 2001.Page: 248–252.
14. Li, C.S., Yu, P.S., Castelli, V.(1998). A Framework For Mining Sequence Database at Multiple Abstraction Levels. In Proceedings of the 1998 ACM 7th international conference on Information and knowledge management, 1998. Pages:267–272.
15. Shasha, D., & Wang, T. L., (1990). New techniques for best-match retrieval. ACM Transactions on Information Systems, Vol. 8, No 2 April 1990, pp. 140-158.