International Journal of Networked and Distributed Computing, Vol. 3, No. 2 (April 2015), 79-88

Optimal Number of Clusters for Fast Similarity Search Considering Transformations of Time Varying Data

Toshiichiro Iwashita*
Dept. of Information Science, Kyoto Institute of Technology, Goshokaido-cho, Matsugasaki, Sakyo-ku Kyoto 606-8585, Japan

Teruhisa Hochin
Dept. of Information Science, Kyoto Institute of Technology, Goshokaido-cho, Matsugasaki, Sakyo-ku Kyoto 606-8585, Japan
E-mail: hochin@kit.ac.jp

Hiroki Nomiya
Dept. of Information Science, Kyoto Institute of Technology, Goshokaido-cho, Matsugasaki, Sakyo-ku Kyoto 606-8585, Japan
E-mail: nomiya@kit.ac.jp

Abstract

This paper proposes a method of determining the optimal number of clusters dividing the multiple transformations for the purpose of the efficient processing of query against the results of applying the transformations to time series. In this paper, the moving average is used as a transformation for simplicity. The model of query time to the number of clusters is constructed for determining the optimal number of clusters. As the query time could be represented with the concave function of the number of clusters, it is shown that the optimal number of clusters for the best query time can be obtained. The verification experiment confirms the validity of the model constructed. It is revealed that the optimal number of clusters could be determined by the times obtained from a single query execution.

Keywords: Time series, Transformation, Retrieval, Cluster, Optimal number.

*Currently, with Fujitsu Ten Technology Ltd.

Published by Atlantis Press

1. Introduction

As the computer technology has rapidly been developed, the computer could treat a variety of multimedia data such as image stills, pieces of music, and movies. However, it is still difficult to retrieve appropriate data from an enormous amount of data adequately and promptly. The time varying data are included in such data. Change of stock price, change of temperature, change of the position of a cellphone, and changes of the position and the diameter of a typhoon are examples of such data. The former two changes are time series data, in which values change according to time. On the other hand, the latter two ones are spatial as well as temporal. In this paper, time series data are treated because of simplicity.

There are many research efforts of addressing to the problem finding time series similar to a time series. [1-9] The simplest way is the one using the Euclidean distance between the points in k dimensional Euclidean space when a time series is represented with k values as dissimilarity of time series. As k is very large in general, the dimension of this space is very high. When the dimension is very high, curse of dimensionality occurs. This is the phenomena that every point is almost equally far from a point in a high dimensional space. In such space, similarity test does not work well because the distance between any pair of points is almost the same. Reducing dimensions is strongly required. Several dimension reduction methods have been proposed.[2-6] One of these methods is based on the transformation from the time domain to the frequency domain. This method reduces dimensions by using only several low frequency components.[2,3] Although only low frequency components are used, similar time series can be retrieved because approximate tendency of a time series is sufficiently represented with them.

Here, the time series obtained by applying a transformation to time series which are similar to the one obtained by applying the transformation to the retrieval key time series may be required to be obtained. For example, the ten-day moving averages of the sequences of stock prices similar to the one of a retrieval key sequence of stock prices are required to be retrieved. Moving averages are widely used in stock data analysis. They could smooth short term fluctuations of stock prices out, and show the basic trend of stock prices. Rafiei *et al.* have proposed an algorithm of efficiently processing this kind of query.[1] This method uses the multidimensional index such as an R-tree index, which manages multi-dimensional data by using minimum bounding rectangles (MBRs). The method efficiently processes such queries by applying the transformation to MBRs.

Rafiei *et al.* have proposed the efficient algorithm for the queries with multiple transformations as well as the one for those with single transformation.[1] For the query with multiple transformations, a set of transformations T is used. The time series obtained by applying a transformation in T to time series which are similar to the one obtained by applying the transformation to the retrieval key time series are retrieved. The query processing method for the query with multiple transformations divides the transformations in T into several clusters. The transformations in a cluster are managed as an MBR in a multidimensional index. This index is called an *MT index*. The effectiveness of the MT index has experimentally been shown. It has also experimentally been shown that the best retrieval performance can be obtained when a cluster contains six to eight transformations for 24 to 48 transformations. Although the condition for the best retrieval performance has been shown, the underlying mechanism is not clear. It is not guaranteed that the best retrieval performance can be attained under such condition.

This paper proposes a method of determining the optimal number of clusters, at which the fastest retrieval performance can be obtained. The moving average is used as a transformation for simplicity. The model of query time to the number of clusters is constructed by building models of the search time and the postprocessing time. The verification experiment confirms the validity of the model constructed. It is revealed that the optimal number of clusters could be determined by the times obtained from a single query execution.

The remaining of this paper is as follows: Section 2 describes the related works including the ST index and the MT index. The model of query time is constructed in Section 3. Section 4 conducts an experiment for confirming the model constructed. Section 5 confirms the agreement of the model of

query time and the optimal number of clusters obtained from the experiment. Lastly, Section 6 concludes this paper.

2. Related Work

2.1. *Fast similarity search of time series*

For the purpose of deriving patterns from time-series data, similarity of two sequences of data must be judged. There are two major approaches in judging the similarity of the sequences. One compares two sequences directly in a time domain.[4-6] The Adaptive Piece-wise Constant Approximation (APCA)[4] approximates a sequence into a sequence of steps, and uses the areas between the steps and the axis. The Multi-resolution Vector Quantized (MVQ)[5] divides a sequence into segments, and approximates it by using a sequence of the representative values of the segments. The ratio of the representative values of a candidate sequence to those of a key one is their similarity. These methods, however, could not properly handle the sequence slightly different from the one in the time direction. For example, the similarity of a sequence and the slightly shifted one becomes very low. In order to adapt this situation, the dynamic programming approach is often taken. The Dynamic Time Warp (DTW)[6] is a popular method.

The other approach compares two sequences in the frequency domain.[2,3,8,9] The sequence of data is transformed to a set of data in the frequency domain. The Fourier transformation is often used as such a transformation. It has been shown that similar sequences can be retrieved by using a few Fourier coefficients.[2] This also brings the dimensionality reduction. We can construct an index for fast retrieval by using only these coefficients.[3,8,9]

By applying Fourier transformation to a sequence of data, Fourier coefficients are obtained. These except for the *0*th one are complex numbers, which is composed of a real number and an imaginary one, while the 0^{th} coefficient is a real number. The *0*th coefficient represents the mean value of the data in the sequence. By using these coefficients, we could compare two segments in the frequency domain. The coefficients of low orders correspond to those of low frequencies, while those of high orders correspond to those of high frequencies. By using coefficients of several lowest orders (all of orders, respectively), we could approximately (precisely) compare two segments.

2.2. *Multi-dimensional index structure*

When the first k Fourier coefficients are used as feature values of a time series, it is represented with a point in a $(2k - 1)$-dimensional space. These points are usually managed by using a multidimensional index such as the R-tree family[10] for the efficient query processing. An R-tree index is constructed by using Minimum Bounding Rectangles (MBRs) including their descendent MBRs. A non-leaf node of an R-tree index structure is the form of *(MBR, ptr)*, where *MBR* is the MBR of all the entries in a child node, and *ptr* is the pointer to the child node. A leaf node is the form of *(MBR, oid)*, where *oid* is the identifier of an object in a database, and *MBR* is the MBR of the object. When an object is a point, its MBR is also a point because a point has no size.

MBRs at the same tree level may overlap. This overlap degrades the retrieval performance. An R*-tree index structure tries to minimize the overlap of MBRs by using several criteria.[10]

2.3. *Treatment of transformation*

Here, the methods proposed by Rafiei *et al.* [1] are described. After the transformations treated are described, the query processing for single transformation and that for multiple transformations are explained.

2.3.1. *Transformation*

Transformations treated by Rafiei *et al.* are limited to scaling and translation,[1] while affine transformations also include rotation and shear.

A transformation is represented with a pair $(\boldsymbol{a}, \boldsymbol{b})$, where $\boldsymbol{a}$ specifies a stretch and $\boldsymbol{b}$ represents a translation. Additionally, applying a transformation $t=(\boldsymbol{a}, \boldsymbol{b})$ to a sequence $\boldsymbol{x}$ is represented with $t(\boldsymbol{x})$. This application follows the calculation of Eq. (1).

$$t(\boldsymbol{x}) = Conv(\boldsymbol{a}, \boldsymbol{x}) + \boldsymbol{b} \quad (1)$$

Here, $Conv(\boldsymbol{x}, \boldsymbol{y})$ means the convolution of $\boldsymbol{x}$ and $\boldsymbol{y}$. The jth element of the convolution of $\boldsymbol{x}$ and $\boldsymbol{y}$, whose length is n, is given by Eq. (2).

$$Conv(\boldsymbol{a}, \boldsymbol{x})_j = \sum_{k=0}^{n-1} x_k \, y_{j-k} \qquad j=0, 1, \cdots, n\text{-}1 \quad (2)$$

Eq. (1) represents the transformation in the time domain. In the frequency domain, the transformation $\boldsymbol{X}$ is represented by Eq. (3) where $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{X}$ are the representations in the frequency domain of $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{x}$, respectively.

$$t(\boldsymbol{X}) = \boldsymbol{A} * \boldsymbol{X} + \boldsymbol{B} \quad (3)$$

Here, $\boldsymbol{X} * \boldsymbol{Y}$ is the element-to-element multiplication of two vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$. Eq. (3) is more easily computed than Eq. (1).

Lastly, we describe the representation of the mean average with the form of $(\boldsymbol{a}, \boldsymbol{b})$. The m mean average of a series, whose length is n, $t_{mavg}=(\boldsymbol{a}, \boldsymbol{b})$ is represented with $\boldsymbol{a}$ and $\boldsymbol{b}$ of Eq. (4).

$$\boldsymbol{a} = [\underbrace{\tfrac{1}{m}, \tfrac{1}{m}, \cdots, \tfrac{1}{m}}_{\mathrm{m}}, 0, 0, \cdots, 0]_{\mathrm{n}}, \quad \boldsymbol{b} = \boldsymbol{0} \quad (4)$$

Here, $\boldsymbol{0}$ is zero vector whose size is n.

2.3.2. *Single transformation*

Next, the method of the query processing with a transformation applied to time series is described. Let us consider the following proximity query as a typical one.

Query 1. Given a query point $\boldsymbol{q}$, a transformation t and a threshold ϵ, find all points $\boldsymbol{x}$ in the data set such that the Euclidean distance $D(t(\boldsymbol{x}), \boldsymbol{q}) < \epsilon$.

The method proposed by Rafiei *et al.* uses an R-tree index. [1] A time series is transformed to in the frequency domain by using Fourier Transformation. The first k Fourier coefficients are inserted into the R-tree index as a point. Query is processed to the R-tree index built in this way. The algorithm of processing Query 1 is as follows:

Algorithm 1.

(i) Preprocessing:

(a) Transform t and $\boldsymbol{q}$ into the frequency domain if they are in the time domain. Let us denote the first k Fourier coefficients of t and $\boldsymbol{q}$ by t_k and $\boldsymbol{q}_k$, respectively.

(b) Build a search rectangle $\boldsymbol{q}_{rect}$ for $\boldsymbol{q}_k$. A search rectangle is the minimum bounding rectangle that contains all points within the Euclidean distance ϵ of $\boldsymbol{q}$. Building a search rectangle is straightforward in the rectangular coordinate system; it is simply $(\boldsymbol{q} - \epsilon, \boldsymbol{q} + \epsilon)$ for $i = 1 \ldots 2k$.

(ii) Search:

(c) If N is not a leaf, apply t to every (rectangle) entry of N and check if the resulting rectangle overlaps $\boldsymbol{q}_{rect}$. For all overlapping entries, call Search on the index whose root node is pointed to by the overlapping entry.

(d) If N is a leaf, apply t to every (point) entry of N and check if the resulting point overlaps $\boldsymbol{q}_{rect}$. If so, the entry is a candidate.

(iii) Post-processing:

(e) For every candidate point $\boldsymbol{x}$, check its full database record to determine if the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{q}$ is at most ϵ. If so, the entry is in the answer set.

2.3.2. *Multiple transformation*

Next, using a set of transformations T is considered. Let us consider the following proximity query to an R-tree index built for a data set S:

Query 2. Given a query time series $\boldsymbol{q}$ and a set T of transformations, find every time series s in S and transformation t in T such that the Euclidean distance $D(t(\boldsymbol{x}), t(\boldsymbol{q})) < \epsilon$.

This query is processed by using the following algorithm:

Algorithm 2. Given an R-tree index which is built on the first k Fourier coefficients of time series and whose root is N, a set of transformations T, a threshold ϵ, and a search sequence $\boldsymbol{q}$, use the index to find all sequences that become within distance ϵ of $\boldsymbol{q}$ after being transformed by a member of T.

(i) Decompose T into sets T_1, T_2, ... using a clustering algorithm.

(ii) Do Steps (iii) to (vi) for every set T_i:
(iii) Build an MBR for points in T_i and project it into a mult-MBR and an add-MBR, which are the MBRs for $\boldsymbol{a}$ and $\boldsymbol{b}$ of a transformation, respectively.
(iv) If N is not a leaf, apply the mult-MBR and the add- MBR to every (rectangle) entry of N and check if the resulting rectangle intersects $\boldsymbol{q}_{rect}$. For every intersecting entry, go to Step (iv) and do this step on the index rooted at the node of the intersecting entry.
(v) If N is a leaf, apply the mult-MBR and the add-MBR to every (point) entry of N and check if the resulting rectangle intersects $\boldsymbol{q}_{rect}$. If so, the entry is a candidate.
(vi) For every candidate entry, retrieve its full database record, apply all transformations in T_i to the sequence, and determine transformations that reduce the Euclidean distance between the data sequence and the query sequence to less than ϵ.

3. Model of Query Time

The model of query time is constructed for obtaining the optimal number of clusters. The transformations treated are limited to the moving average. A cluster is assumed to be created by merging transformations one by one. Each cluster has the same number of transformations in it.

3.1. *Models of times*

3.1.1. *Query time*

The time treated here is the time in processing Query 2 by using an MT index. The time of creating clusters is not considered. That is, the time t_{query}, which is taken from Step (iv) to Step (vi) of Algorithm 2, is considered. Here, Step (iv) and Step (v) are for searching candidates, while Step (vi) confirms whether they truly satisfy the retrieval condition, which is the post-processing. The times required for them are referred to as t_{search} and $t_{postprocessing}$, respectively. The time of processing a query t_{query} is the sum of t_{search} and $t_{postprocessing}$.

3.1.2. *Search time*

In the query processing using an MT index, retrieval is repeated N_C times, where N_C is the number of clusters. It is assumed that the retrieval time $t_{search\_on\_index}$ is constant. As the retrieval time t_{search} is proportional to N_C, t_{search} is obtained by Eq. (5).

$$t_{search} = N_C \cdot t_{search\_on\_index} \tag{5}$$

3.1.3. *Post-processing time*

In Step (vi) of Algorithm 2, for each time series retrieved from each cluster, the confirmation is repeated N_i times, where N_i is the number of transformations in the ith cluster. The number of these confirmations is called the number of candidates $N_{candidate}$. Let the time of confirming a candidate be denoted as $t_{confirm\_answer}$. The post-processing time $t_{postprocessing}$ is obtained by Eq. (6).

$$t_{postprocessing} = N_{candidate} \cdot t_{confirm\_answer} \tag{6}$$

3.2. *Number of candidates*

It is assumed that data are uniformly distributed. The number of candidates $N_{candidate}$ is proportional to the ratio of the rectangle R obtained by applying the transformation rectangle to a time series to the whole area R_{all}, which is the MBR obtained by applying the transformation rectangles in a cluster to a time series.

Let N_T be the total number of transformations. The number of transformations in a cluster is N_T/N_C because it is assumed that each cluster has the same number of transformations. For each dimension, the width of the retrieval rectangle is considered to be decided by the number of transformations in a cluster. For the number of dimensions n, the volume V_R of the hyper-rectangle R obtained by transformation is obtained in Eq. (7).

$$V_R = \alpha_1 \cdot \left(\frac{N_T}{N_C}\right)^n \tag{7}$$

Here, α_1 is a constant.

Let N_D be the number of data. There are $N_D \cdot N_C$ rectangles such as R. When overlaps of Rs could be ignored, the sum V_{Rsum} of the volumes V_R is obtained by Eq. (8).

$$\begin{aligned} V_{R_{sum}} &= V_R \cdot N_D \cdot N_C \\ &= \alpha_1 \cdot \left(\frac{N_T}{N_C}\right)^n \cdot N_D \cdot N_C \\ &= \alpha_1 \cdot N_D \cdot N_T^n \cdot N_C^{1-n} \end{aligned} \quad (8)$$

Here, the total number of candidates is $N_D \cdot N_C$. Assume that the probability of a hit is represented with $\alpha_2 \cdot V_{Rsum}$ by using some constant α_2. The number of candidate $N_{candidate}$ is obtained by Eq. (9).

$$\begin{aligned} N_{candidate} &= \alpha_2 \cdot V_{R_{sum}} \cdot N_D \cdot N_T \\ &= \alpha_2 \cdot \alpha_1 \cdot N_D \cdot N_T^n \cdot N_C^{1-n} \cdot N_D \cdot N_T \\ &= \alpha_3 \cdot N_C^{1-n} \end{aligned} \quad (9)$$

Here, $\alpha_3 = \alpha_1 \alpha_2 N_T^{n+1} N_D^2$

3.3. *Optimal number of clusters*

The retrieval time t_{query} is obtained by Eq. (10).

$$t_{query} = N_C \cdot t_{search\_on\_index} + \alpha_3 \cdot N_C^{1-n} \cdot t_{confirm\_answer} \quad (10)$$

Let the retrieval time t_{query} of Eq. (10) be a function of the number of clusters N_C. This is because $t_{search\_on\_index}$ and $t_{confirm\_answer}$ are considered to be constant, and α3 is a constant. As the function $t_{query}(N_C)$ is a concave function, this function has a minimum value. When $dt_{query}/dN_C = 0$, Eq. (11) is obtained.

$$N_C^{-n} = \frac{t_{search\_on\_index}}{\alpha_3 \cdot (n-1) \cdot t_{confirm\_answer}} \quad (11)$$

As N_C is larger than zero, N_C is obtained by Eq. (12).

$$N_C = \sqrt[n]{\alpha_3 \cdot (n-1) \cdot \frac{t_{confirm\_answer}}{t_{search\_on\_index}}} \quad (12)$$

Eq. (12) says that N_C, which minimizes t_{query}, can be obtained when the number of dimensions n, the time of an index retrieval $t_{search\_on\_index}$, the post-processing time $t_{confirm\_answer}$, and the constant α_3 are obtained.

4. Verification Experiment

The model of query time constructed in the previous section is experimentally confirmed.

4.1. *Experimental method*

The retrieval times are measured 1,000 times on a Linux computer (Intel Xeon E5620, 2.40GHz, 4 core, 8 thread × 2, 16GB memory, CentOS 5.7). The results are evaluated in the average. Unit of time is milliseconds. The experimental environments are as follows:

- Target time series data are the following artificial ones:

$$\boldsymbol{x} = [x_t],\ x_t = x_{t-1} + z_t \quad (13)$$
$$x_0,\ z_t\text{: random values of [-500,500]}$$

  This randomization makes the distribution of data uniform.
- The number of dimensions, the length, and the number of time series are 4, 256, and 6000, respectively.
- The running average is used as the transformation. The following three sets of moving averages are used: 1 to 50 units, 1 to 75 units, and 1 to 100 units. The numbers of transformations are 50, 75, and 100, respectively.
- The numbers of transformations in a cluster are the divisors of the number of transformations. The transformations are sequentially stored into clusters.
- Key time series are randomly selected in R_{all}.
- Following the previous work, [1] ϵ is set according to Eq. (14).

$$\epsilon = \sqrt{2(n-1)(1-\rho)} \quad (14)$$

  Here, n is the length of a time series, and ρ = 0.96.

4.2. *Experimental result*

4.2.1. *Query time and optimal number of clusters*

The retrieval times t_{query}. measured and averaged are shown in Fig. 1. Those against the small N_C, whose range is [0, 30], are shown in Fig. 2 for improving visibility. The retrieval time t_{query} is a concave function as shown in Fig. 1 and Fig. 2. As we can see

in Fig. 2, the retrieval time t_{query} becomes minimum at $N_C = 5$.

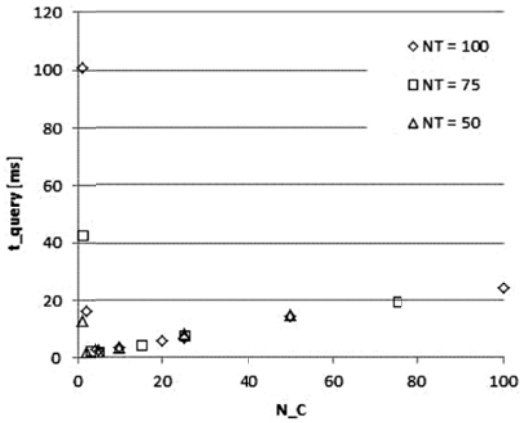

Fig. 1. Retrieval time t_{query} against the number of clusters N_C for the number of transformations N_T = 100, 75, 50.

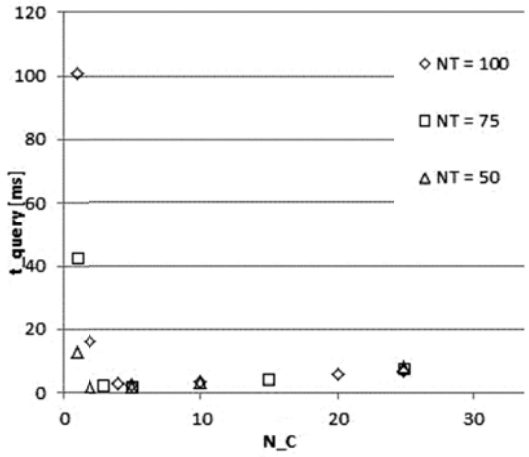

Fig. 2. Retrieval time t_{query} against the small number of clusters N_C ([0, 30]) for the number of transformations N_T=100, 75, 50.

4.2.2. *Query time and optimal number of clusters*

The search times t_{search} are shown in Fig. 3. It can be seen that the search time t_{search} is proportional to the number of clusters N_C.

The search times using an index $t_{search\_on\_index}$ are shown in Table 1. Table 1 includes the middle value. It can be seen that the values except for those at N_C = 1 are around the middle value.

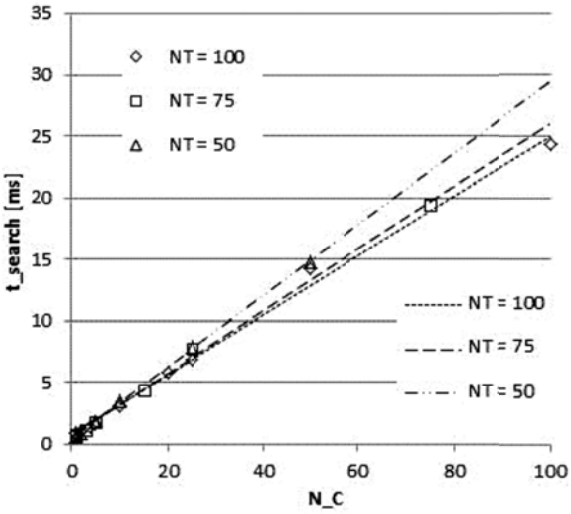

Fig. 3. Search time t_{search} against the number of clusters N_C for the number of transformations N_T = 100, 75, 50 and its approximation curve.

Table 1. Search time using index $t_{search\_on\_index}$ [ms]

N_C	$N_T = 100$	$N_T = 75$	$N_T = 50$
100	*0.244*	-	-
75	-	*0.260*	-
50	*0.284*	-	*0.296*
25	*0.275*	*0.308*	*0.318*
20	*0.289*	-	-
15	-	*0.287*	-
10	*0.313*	-	*0.343*
5	*0.329*	*0.349*	*0.355*
4	*0.351*	-	-
3	-	*0.377*	-
2	*0.487*	-	*0.445*
1	*0.921*	*0.789*	*0.631*
Middle value	*0.313*	*0.328*	*0.349*

4.2.3. *Query time and optimal number of clusters*

The post-processing time $t_{postprocessing}$ is shown in Fig. 4. It can be seen that the post-processing time $t_{postprocessing}$ decreases according to the number of clusters N_C.

The sum of the volumes of hyper-rectangles V_{Rsum} is shown in Fig. 5. Here, the one is omitted when $N_C = N_T$ because V_{Rsum} is equal to zero. Fig. 5 includes approximation curves, which are based on the power function. It is considered that V_{Rsum} is proportional to the power of -3.5 of N_C.

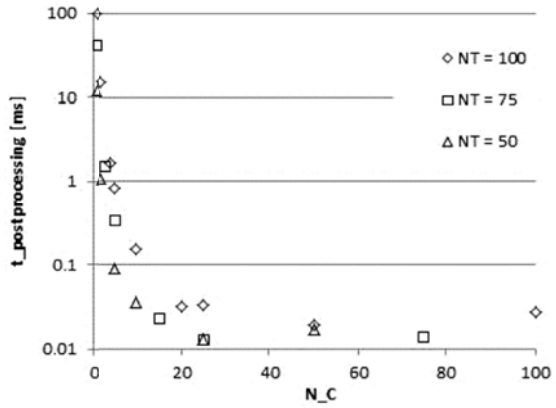

Fig. 4. Post-processing time $t_{postprocessing}$ against the number of clusters N_C for the number of transformations N_T = 100, 75, 50.

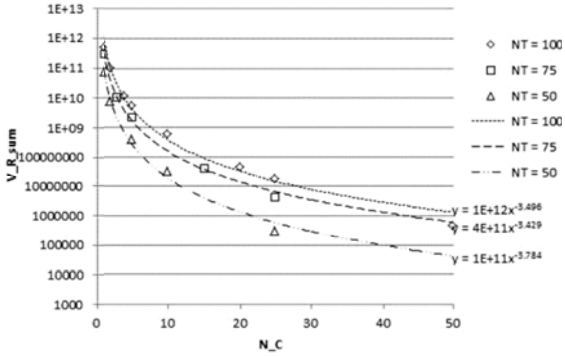

Fig. 5. Sum of the volumes of hyper-rectangles V_{Rsum} against the number of clusters N_C for the number of transformations N_T = 100, 75, 50 and its approximation curve.

Table 2. Number of candidates $N_{candidate}$

N_C	$N_T = 100$	$N_T = 75$	$N_T = 50$
100	0.169	-	-
75	-	0.111	-
50	0.252	-	0.109
25	0.576	0.273	0.170
20	0.805	-	-
15	-	0.650	-
10	3.13	-	0.570
5	18.8	7.68	2.06
4	37.8	-	-
3	-	34.35	-
2	352.4	-	26.0
1	2482	1183	293.8

The numbers of candidates $N_{candidate}$ are shown in Table 2. The values obtained by subtracting the minimum values from the original ones are shown in Fig. 6. Fig. 6 also includes approximation curves based on the power function. It can be seen that the number of candidates $N_{candidate}$ is proportional to the power of -2.7.

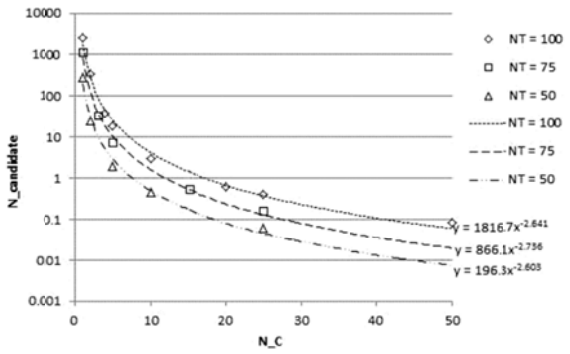

Fig. 6. The number of candidates $N_{candidate}$ against the number of clusters N_C for the number of transformations N_T = 100, 75, 50 and its approximation curve.

The confirmation time $t_{confirm\_answer}$ is shown in Table 3. Table 3 also includes middle values. It can be seen that the values except for those at $N_C = N_T$ are around the middle values.

Table 3. Confirmation time $t_{confirm\_answer}$ [ms]

N_C	$N_T = 100$	$N_T = 75$	$N_T = 50$
100	0.160	-	-
75	-	0.126	-
50	0.075	-	0.156
25	0.057	0.048	0.076
20	0.040	-	-
15	-	0.035	-
10	0.049	-	0.061
5	0.043	0.045	0.044
4	0.044	-	-
3	-	0.044	-
2	0.043	-	0.415
1	0.040	0.035	0.042
Middle value	0.046	0.045	0.061

Table 4. The times $t_{search\_on\_index}$ and $t_{confirm\_answer}$, and the constant α_3 obtained.

N_T	$t_{search\_on\_index}$ [ms]	$t_{confirm\_answer}$ [ms]	α_3
50	0.343	0.061	570.0
75	0.287	0.035	2194
100	0.289	0.040	6440

5. Evaluation

5.1. *Time and constant*

5.1.1. *Search time*

The search time t_{search} is proportional to the number of candidates N_C as shown in Fig. 3. Therefore, it is confirmed that Eq. (5) is valid.

5.1.2. *Post-processing time*

The sum of the volumes of hyper-rectangles V_{Rsum} is proportional to the power of -3.5 of N_C as shown in Fig. 5. It is considered that the tendency of V_{Rsum} is quite similar to that of Eq. (8).

The number of candidates $N_{candidate}$ is proportional to the power of -2.7 as shown in Fig. 6. It is also considered that the tendency of $N_{candidate}$ is quite similar to that of Eq. (9).

When we use the confirmation times $t_{confirm\_answer}$ at $N_C = N_T / 5$ shown in Table 3, the typical times may be obtained for every N_T. The confirmation times $t_{confirm\_answer}$ obtained are 0.061[ms], 0.035[ms], and 0.040[ms] for N_T = 50, 75, 100, respectively.

5.1.3. *Constant*

The constant α_3 is represented with $N_{candidate} \cdot N_c^{n-1}$ based on Eq. (9). The constant α_3 is obtained by using the number of candidates $N_{candidate}$, shown in Table 2, at $N_C = N_T/5$, as in the retrieval time and the postprocessing one. The constants α_3 obtained are 570.0, 2193.8, and 6440.0 at N_T = 50, 75, 100, respectively.

The times $t_{search\_on\_index}$ and $t_{confirm\_answer}$, and the constant α_3 obtained up to here are shown in Table 4.

5.2. *Optimal number of clusters*

The optimal numbers of clusters N_{C50}, N_{C75}, and N_{C100} obtained by using Eq. (12) in four dimensions are 4.2, 5.3, and 7.2, respectively. Therefore, the query time is the minimum when N_C is equal to 5 for all of the numbers of transformations: 50, 75, and 100.

On the other hand, it can be seen that the query time is the minimum when N_C is equal to 5 in the experiment as shown in Fig. 1 and Fig. 2.

As we have seen, both of the model of query time and the experimental result show that the optimal number of clusters N_C is 5. Moreover, Eq. (10) is agreed with the experimental result shown by Rafiei *et al.*[1]

The optimal number of clusters N_C can be obtained by using Eq. (12) with $t_{search\_on\_index}$, $t_{confirm\_answer}$, and α_3 obtained under the condition that $N_C = N_T / 5$.

6. Conclusion

This paper proposed a method of determining the optimal number of clusters dividing the multiple transformations for the purpose of the efficient processing of query against the result of applying the transformations to time series. In this paper, the moving average is used as a transformation for simplicity. The model of query time to the number of clusters was constructed by building models of search time and postprocessing time for determining the optimal number of clusters. The verification experiment confirmed the validity of the model constructed. It was revealed that the optimal number of clusters could be determined by the times obtained from a single query execution.

In this paper, the experiment is conducted under the fixed environment: the fixed data sets, the fixed number of transformations, and the fixed dimensions. Experiments under other conditions are required to be conducted. This paper treats only the moving average as a transformation. Confirming the validity of the model for other kinds of transformation including time shift is also in future work. The model is also required to be confirmed when other clustering methods are adopted. In this paper, time series data are treated as time varying ones. The value changes according to the time in a time series. There are time varying data such that the position and/or the size in two or three dimensional space may change according to the time. Treating the time varying data whose changes are spatial as well as temporal is in future work.

References

1. D. Rafiei and A. O. Mendelzon, Querying time series data based on similarity, *IEEE Transactions on Knowledge and Data Engineering*, 12(5) (2000), pp. 675-693.

2. R. Agrawal, C. Faloutsos, and A. N. Swami, Efficient similarity search in sequence databases, in *Proc. 4th Int'l Conf. on Foundations of Data Organization and Algorithms (FODO'93)* (1993), pp. 69-84.
3. C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, Fast subsequence matching in time-series databases, in *Proc. 1994 ACM SIGMOD Int'l Conf. on Management of Data* (1994), pp. 419-429.
4. E. J. Keogh, K. Chakrabarti, S. Mehrotra, and M. J. Pazzani, Locally adaptive dimensionality reduction for indexing large time series databases, in *Proc. 2001 ACM SIGMOD Int'l Conf. on Management of Data* (2001), pp. 151-162.
5. V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos, Multiresolution Symbolic Representation of Time Series, in *Proc. 21st IEEE Int'l Conf. on Data Engineering* (2005), pp. 668-679.
6. S.-W. Kim, S. Park, and W. W. Chu, An Index Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases, in *Proc. 17th Int'l Conf. on Data Engineering* (2001), pp. 607-614.
7. T. Hochin, K. Koyama, H. Nakanishi, M. Kojima, and LABCOM group, Extension of frequency-based dissimilarity for retrieving similar plasma waveforms, *Fusion Engineering and Design*, 83(2) (2008), pp. 417-420.
8. T. Hochin, Y. Yamauchi, H. Nomiya, H. Nakanishi, and M. Kojima, Fast subsequence matching in plasma waveform databases, in *Proc. 5th Int'l Conf. on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP2009)* (2009), pp. 759-762.
9. T. Hochin, Y. Yamauchi, H. Nakanishi, M. Kojima, and H. Nomiya, Indexing of plasma waveforms for accelerating search and retrieval of their subsequences, *Fusion Engineering and Design*, 85(5) (2010), pp. 649-654.
10. N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, The R*-tree: an efficient and robust access method for points and rectangles, in *Proc. 1990 ACM SIGMOD Int'l Conf. on Management of Data* (1990), pp. 322-331