International Journal of Innovative
Computing, Information and Control
Volume **12**, Number **2**, April **2016**

ICIC International ©2016 ISSN 1349-4198
pp. 419–431

COEFFICIENT CONTROL MULTI-STEP *k*-NN SEARCH IN TIME-SERIES DATABASES

Sanghun Lee[1], Bum-Soo Kim[2], Mi-Jung Choi[1] and Yang-Sae Moon[1,*]

[1]Department of Computer Science
Kangwon National University
No. 1, Kangwondaehak-gil, Chunchon, Kangwon 24341, Republic of Korea
{ sanghun; mjchoi }@kangwon.ac.kr; *Corresponding author: ysmoon@kangwon.ac.kr

[2]Department of Knowledge Science Engineering
Korea Advanced Institute of Science and Technology
KAIST 291, Daehak-ro, Yuseong, Daejeon 34141, Republic of Korea
bskim@kangwon.ac.kr

Received September 2015; revised January 2016

Abstract. *The k-NN search is widely used for similarity search on various time-series data such as image, text, trajectory, and biomedical data. In this paper, we address the problem of improving the performance of multi-step k-NN search on a multidimensional index. The existing multi-step k-NN search has a critical performance problem: it produces a large tolerance from a k-NN query on the index due to use of dimensionality reduction, and the large tolerance incurs a large number of candidates, which lead to severe I/O and CPU overhead. To overcome this problem, we propose a new solution, called coefficient control multi-step k-NN search (cc-kNN search in short), which uses $c \cdot k$ instead of k in a k-NN query to obtain a tight tolerance. For this, we intuitively explain why a simple operation of increasing k can produce the tight tolerance and formally prove that the cc-kNN search finds k results correctly without any false dismissal. We also define the control constant c used in the k-NN query and formally present how to construct an estimation function for determining the constant c. Experimental results show that the proposed cc-kNN search beats the existing multi-step k-NN search in the execution time as well as the number of candidates.*

Keywords: Multi-step k-NN search, Similarity search, Time-series data, Dimensionality reduction, Multi-dimensional index

1. **Introduction.** Due to advances in IT technologies, the amount of time-series data collected from social networks, scientific experiments, business applications, and spatio-temporal services is explosively increased. To effectively store and analyze those time-series data, there have been many efforts on *similarity search* (or *matching*) on a large volume of time-series data [9, 10, 20, 30]. As representative matching techniques for time-series data, range search and k-NN search have been popularly used in many applications [11, 12, 23, 24]. In particular, the k-NN search has been widely applied for not only simple similarity search but also various applications such as recommender systems [22] and graph databases [2]. Thus, the k-NN search is a very important issue in similarity search on large time-series databases.

In this paper, we focus on the performance improvement of *multi-step k-NN search* [14] in time-series databases. The multi-step k-NN search has many applications in multimedia databases, especially in image databases. Its examples include video similarity search [29], which uses earth movers distances in the multi-step search, and matrix similarity search

[8], which identifies similar sub-windows from large-scale images by exploiting the multi-step search. In general, the multi-step k-NN search, which identifies similar time-series (or sequences) using a multidimensional index, generally consists of two major steps. The first step, *tolerance determination*, performs a k-NN query on the index of low dimensional sequences, computes the actual (high dimensional) distances from the query sequence to the retrieved k data sequences, and sets the k-th distance to a tolerance to be used in a range query of the second step. The second step, *candidate filtering*, makes a range query by using the tolerance as its search range, retrieves candidate data sequences by executing the range query on the index, and obtains the final k data sequences by removing false alarms from the candidates. This multi-step k-NN search, however, has a critical problem of incurring too many candidates in the second step since the first step often produces a large tolerance. The more candidates it has, the heavier I/O and CPU overhead it incurs to filter out false alarms from those many candidates. Thus, reducing the number of candidates is a very important factor in improving the performance of multi-step k-NN search.

To improve the performance of multi-step k-NN search, we first analyze why there are so many candidates. The analysis result says that this is due to information loss by use of dimensionality reduction. To use the multidimensional index, the multi-step k-NN search exploits the dimensionality reduction for transforming high dimensional sequences to low dimensional sequences. In the tolerance determination step, we retrieve k low dimensional sequences by executing a k-NN query on the index. The retrieved k sequences, however, may not be actual k results since the dimensionality reduction usually causes information loss. Thus, we compute actual high dimensional distances for the retrieved k sequences and execute a range query on the index by setting its search range to the largest distance (i.e., the k-th distance). Then, the range query produces candidate sequences which contain false alarms as well as actual k nearest sequences. Here we note that the larger tolerance we use, the more candidates we investigate. That is, a large tolerance in the tolerance determination step incurs a large number of candidates in the candidate filtering step.

To overcome the problem of many candidates in the range query, in this paper we propose a new search method, called *coefficient control multi-step k-NN search* (*cc-kNN search* in short)[1]. The cc-kNN search simply increases k to $c \cdot k$ ($c > 1$) for the k-NN query on the index. Surprisingly, this simple change produces a tight tolerance and eventually improves the performance by reducing the candidates. For this, we first formally prove the correctness of the cc-kNN search, i.e., it finds the most similar k sequences correctly without any false dismissal. We then formally define the *control constant* c, which is used for determining the increment of k, and the *c-estimation function*, which is used for estimating c. We also present a formal method of constructing an appropriate estimation function by using the regression analysis [5, 6]. Experimental results show that the proposed *cc-kNN* search reduces the number of candidates and the search time compared to the existing multi-step k-NN search. Therefore, we believe that our cc-kNN search is an excellent solution that improves the performance of multi-step k-NN search without changing its basic structure and intrinsic working mechanism.

The rest of this paper is organized as follows. Section 2 explains the related work. Section 3 analyzes the problem of the previous multi-step k-NN search. Section 4 presents the proposed cc-kNN search with the concept, the control constant determination, and

[1]The preliminary Korean version of this paper was published in *Journal of KIISE*, vol.42, no.2, pp.242-254, 2015. This is an extended and formalized version of that paper.

the estimation function derivation. Section 5 empirically shows the superiority of the cc-kNN search. We finally summarize and conclude the paper in Section 6.

2. Related Work.

2.1. Similarity search on time-series data.

Time-series data are the sequences of real numbers representing values at specific time points [1, 9, 16]. The time-series data stored in a database are called *data sequences*. Finding data sequences similar to the given *query sequence* is called *similarity search* or *similar sequence matching*, which is classified into range search and k-NN search. The range search finds data sequences whose distances from the query sequence are less than or equal to the user-specified tolerance ϵ. The k-NN search finds k nearest data sequences to the query sequence from the time-series database. In this paper, we focus on the k-NN search which has been much more widely used than the range search.

To improve the performance of similarity search in time-series databases, many research efforts use multidimensional indexes [7, 9, 15, 17, 19]. By using the index, we can significantly reduce the execution time compared with the sequential scan. However, storing high dimensional sequences directly in the index causes the high dimensionality problem, called curse of dimensionality [4]. To avoid this problem, many previous works [1, 9, 11, 16, 20] exploit the dimensionality reduction, that is, they first transform high dimensional sequences to low dimensional sequences and then store those low dimensional sequences into a multidimensional index.

Dimensionality reduction [9, 15] is the process of converting high dimensional sequences into low dimensional sequences in order to use multidimensional indexes effectively. To use the dimensionality reduction in similarity search, however, the high and low dimensional distances should satisfy Lemma 2.1, which is also known as Parseval's theorem [1, 9].

Lemma 2.1. *To guarantee no false dismissal in similarity search, the dimensionality reduction should satisfy Equation (1) for high dimensional sequences X, Y and their corresponding low dimensional sequences X', Y'.*

$$D_{real}(X, Y) \geq D_{feature}(X', Y') \tag{1}$$

In Equation (1), $D_{real}(\cdot)$ is a distance function for high dimensional sequences, and $D_{feature}(\cdot)$ is the one for low dimensional sequences.

Proof: Readers are referred to [9, 15]. □

2.2. Multi-step k-NN search.

As we explained in Section 1, the multi-step k-NN search consists of two steps: tolerance determination and candidate filtering steps. Figure 1 shows the two-step procedure of multi-step k-NN search by Korn et al. [14]. As shown in the figure, each step of the multi-step k-NN search is composed of two subparts.

- *Tolerance determination*: In Part (a), it performs a k-NN query on the index and obtains k low dimensional sequences as primary candidates. In Part (b), it computes high dimensional distances from those candidates to the query sequence Q by accessing the database and sets the tolerance, $d_{\max}$, to the largest distance (i.e., k-th distance) among those k high dimensional distances.
- *Candidate filtering*: In Part (c), it first constructs a range query using the tolerance ($d_{\max}$) of Part (b) as the search range, and by executing the range query on the index, it then obtains final candidates whose distances are within the tolerance. In Part (d), it computes high dimensional distances from those final candidates to the query sequence and reports the k nearest data sequences as the final result.

Lemma 2.2 shows the correctness of the multi-step k-NN search.

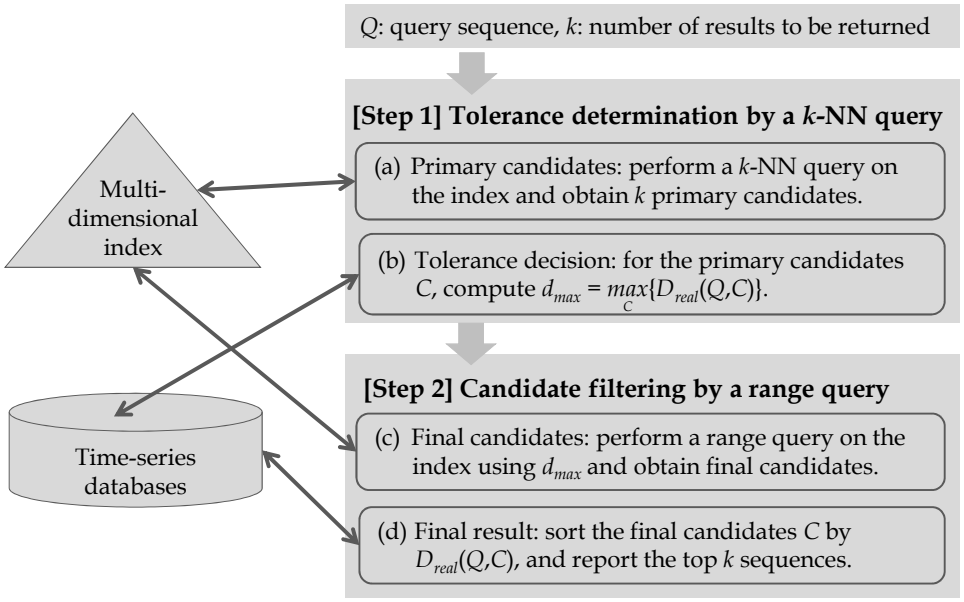


FIGURE 1. Procedure of multi-step k-NN search

Lemma 2.2. *The multi-step k-NN search consisting of tolerance determination and candidate filtering steps correctly finds the k nearest data sequences to the query sequence without any false dismissal.*

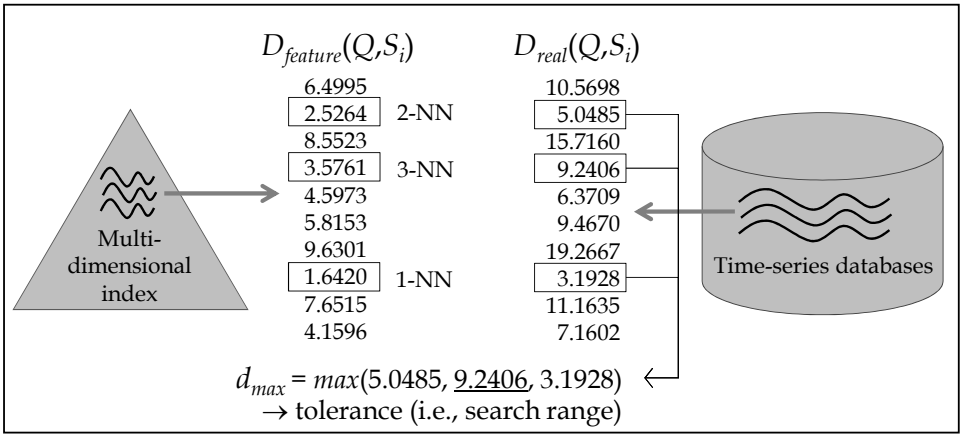
Proof: Readers are referred to [14]. □

There have been a few efforts on improving the performance of multi-step k-NN search. Tao et al. [28] propose the LSH (locality sensitive hashing)-based B-tree for the performance improvement. Seidl and Kriegel [26] improve the performance by using q-ranking [27] and GEMINI (generic multimedia indexing) [9] methods step by step instead of the existing two steps of multi-step k-NN search. However, Tao et al.'s solution has a problem in using the general purpose index such as R*-tree [3] since it does not enhance the multi-step framework itself but does change an index structure for the special purpose. Also, Seidl and Kriegel's solution has a problem of not following the basic structure of original multi-step k-NN search. Therefore, in this paper we propose an improved solution of multi-step k-NN search which follows the basic structure itself and at the same time uses a general purpose index for the easy implementation.

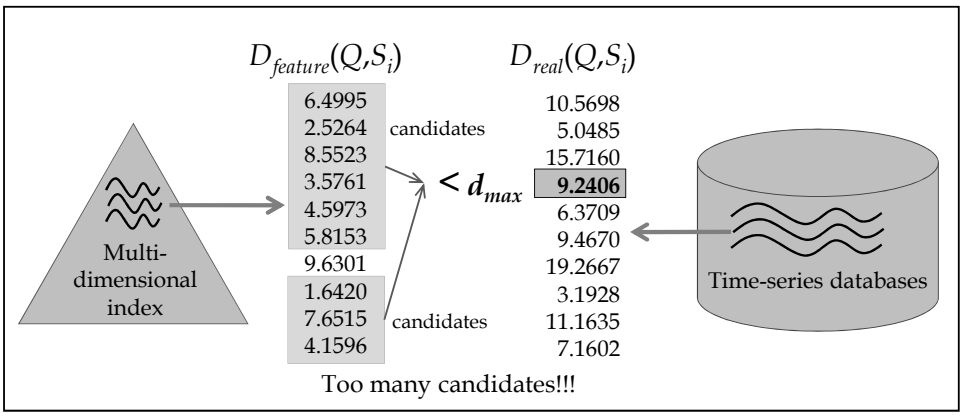
3. Problem Analysis on Multi-Step k-NN Search.

In this section, we analyze the reason "why so many candidates occur in a range query of the multi-step k-NN search" in detail, which motivates the paper. The direct reason of many candidates is a large search range of the range query. In other words, the reason is that the tolerance obtained in the tolerance determination step is large. Then, why is the tolerance too large? This is because we use the dimensionality reduction for the index. More precisely, the tolerance obtained in the tolerance determination step is a high dimensional distance, but the range query in the candidate filtering step uses that tolerance as a low dimensional distance on the index. In general, the high dimensional distance is much larger than the low dimensional distance, and thus, the discrepancy between these high and low dimensional distances produces many candidates as the result of a range query.

Example 3.1 shows a real case of explaining why many candidates occur by the dimensionality reduction.



(a) Step 1: tolerance determination



(b) Step 2: candidate filtering

FIGURE 2. Example of incurring many candidate sequences by the dimensionality reduction

Example 3.1. *Figure 2 shows a real example of performing a multi-step 3-NN search query. As shown in Figure 2(a), we first find three candidates, 3-NN in the figure, by searching the index in the tolerance determination step. We then determine the tolerance as 9.2406 by computing the high dimensional distances from those 3-NN data sequences to the query sequence. Here we note that this tolerance is much larger than the low dimensional distance in between query and data sequences. As shown in Figure 2(b), all low dimensional distances except one (= 9.6301) are less than the tolerance (= 9.2406) determined in Figure 2(a). Thus, if we perform a range query by using 9.2406 as its search range, nine data sequences, 90% of data sequences, are determined as the candidates as shown in the left side of Figure 2(b).*

As shown in Example 3.1, the major reason of performance degradation in multi-step k-NN search is the discrepancy between high and low dimensional distances. That is, to guarantee no false dismissal, we need to determine the tolerance as a high dimensional distance, and this high dimensional distance incurs many candidates in a low dimensional range query on the low dimensional index.

Based on this observation, in this paper we propose a coefficient control approach which tries to reduce the tolerance of a range query. The basic strategy of the proposed approach is that the more effort in the tolerance determination step we do, the less overhead in the candidate filtering step we have. More precisely, we investigate the larger number of candidates in the tolerance determination step for the purpose of obtaining the tolerance

as small as possible, and by using the small tolerance as the search range of a range query, we can reduce the number of candidates in the candidate filtering step.

4. **Coefficient Control Multi-Step *k*-NN Search.** In this section, we propose the coefficient control multi-step k-NN search, cc-kNN search, which obtains a tight tolerance by simply replacing k as $c \cdot k$ in a k-NN query on the index. In Section 4.1, we explain the concept of the proposed cc-kNN search. In Section 4.2, we present a formal method of determining the control constant c and show its example.

4.1. **The concept.** The cc-kNN search is based on an intuition that the more candidates we investigate by increasing k to $c \cdot k$ ($c > 1$), the smaller tolerance we get in the tolerance determination step. In the existing multi-step k-NN search, we retrieve k data sequences by executing a k-NN query and set the largest (i.e., k-th) distance to the tolerance. Then, what happens if we use $c \cdot k$ instead of k? We get $c \cdot k$ data sequences as candidates, and accordingly, we obtain $c \cdot k$ high dimensional distances to the query sequence. Among these $c \cdot k$ high dimensional distances, we choose the k-th one, which is most likely to be smaller than the largest one among the previous k high dimensional distances. Using this smaller tolerance we can eventually improve the performance of multi-step k-NN search. Here we note that replacing k to $c \cdot k$ in the multi-step k-NN search is a very simple modification, and we do not need to change any of basic structures and intrinsic working mechanisms of the original multi-step k-NN search algorithms and multi-dimensional indexes. More importantly, this simple modification significantly improves the overall search performance, and it can maximize the performance of multimedia retrieval applications [8, 12, 24, 29].

To use the cc-kNN search, we need to determine a constant c first, and we define it as follows.

Definition 4.1. *We define c as a control constant, which is used to increase k to $c \cdot k$ in the tolerance determination step of the cc-kNN search, and we determine c by using a c-estimation function $f(k)$ of Equation (2).*

$$c = f(k). \tag{2}$$

The c-estimation function $f(k)$ is a regression equation derived from the nonlinear regression analysis [25]. We determine the function $f(k)$ from a few k values, $k_1, \ldots, k_n$, and their corresponding c values, $c_1, \ldots, c_n$, which are given by the preliminary experiment[2]. We represent these k and c pairs as a set $\mathbb{A}$ of scatter plots as in Equation (3).

$$\mathbb{A} = \{(k_1, c_1), (k_2, c_2), \ldots, (k_n, c_n)\}. \tag{3}$$

In Equation (3), a pair (k_i, c_i) means that, for a given coefficient k_i, if we use a control constant c_i, we can minimize the execution time of the cc-kNN search. Thus, if we have the set $\mathbb{A}$ by the preliminary experiment, we can derive a c-estimation function $f(k)$ by using the regression analysis, and we will explain the detailed procedure in Section 4.2.

Next, to use the set $\mathbb{A}$ of Equation (3), we need to know a control constant c_i for a given k_i. In this paper, we determine it through a preliminary experiment on the real database (or sample database if necessary). Briefly speaking, for a given k_i, we execute the multi-step k-NN search repeatedly using a few constants and choose a one which shows the smallest execution time as c_i. To explain the concept more formally, we define a *k_i-time function*, $g_i(x)$, which determines a control constant c_i for a given coefficient k_i.

[2]The best way to making a function $f(k)$ is to investigate all possible k's and their optimal c's. However, this is not feasible due to heavy computation. Thus, in this paper we derive $f(k)$ from a few (k, c) pairs, which are obtained by the preliminary experiment, and dynamically apply the function for any k to determine its pseudo-optimal c.

Definition 4.2. *For a given coefficient k_i, we define $g_i(x)$ as a k_i-time function which returns the execution time of multi-step k-NN search by using $x \cdot k_i$ instead of k_i in a k-NN query on the index.*

If we know a k_i-time function $g_i(x)$ for a given k_i, we can determine its control constant c_i as the minimum of $g_i(x)$ as in Equation (4).

$$c_i = \min_x \{g_i(x)\}. \tag{4}$$

To derive the k_i-time function $g_i(x)$, we use the nonlinear regression analysis again. For a given k_i, we repeat the multi-step k-NN search by using $1 \cdot k_i$, $2 \cdot k_i$, ..., $m \cdot k_i$ and make a set of scatter plots, $\mathbb{Z}_i$ of Equation (5), by measuring the execution times.

$$\mathbb{Z}_i = \{(1, t_1), (2, t_2), \ldots, (m, t_m)\}. \tag{5}$$

In Equation (5), a pair (j, t_j) means that the multi-step k-NN search takes t_j time if we use $j \cdot k_i$ instead of k_i in a k-NN query on the index. Through a preliminary experiment, we first construct a set $\mathbb{Z}_i$ for a given k_i and then obtain a k_i-time function $g_i(x)$ from $\mathbb{Z}_i$ by the regression analysis. We explain the detailed procedure in Section 4.2.

4.2. **Determination of control constants.** In this section, we explain the detailed procedure of deriving k_i-time and c-estimation functions, which are used for determining appropriate control constants. The procedure consists of the following four steps: (1) construction of a set $\mathbb{Z}_i$, (2) derivation of a k_i-time function $g_i(x)$, (3) derivation of a c-estimation function $f(k)$, and (4) determination of a control constant c. The first three steps (1) to (3) are executed only once as a pre-processing phase, but the last step (4) is executed at each time for a given k to obtain its appropriate control constant c.

(1) **Construction of** $\mathbb{Z}_i$: Determine the coefficients $k_1, \ldots, k_n$, and also determine m, the number of repetitions used for constructing $\mathbb{Z}_i$ for each k_i.[3] After then, for each k_i, construct $\mathbb{Z}_i$ by executing the multi-step k-NN search m times for $1 \cdot k_i$, $2 \cdot k_i$, ..., $m \cdot k_i$.

(2) **Derivation of** $g_i(x)$: Through the regression analysis, derive a k_i-time function, which has a form of a fractional function of Equation (6). The fractional function $g_i(x)$ of Equation (6) is recommended by the regression analysis of SPSS [25].

$$g_i(x) = \frac{\delta_i}{\alpha_i x} + \beta_i x + \zeta_i. \tag{6}$$

(3) **Derivation of** $f(k)$: Set a control constant c_i of k_i to a minimum of $g_i(x)$ as in Equation (4) and construct a set $\mathbb{A}$ of (k_i, c_i) pairs as in Equation (3). Then, perform the regression analysis again for the set $\mathbb{A}$ and derive a c-estimation function $f(k)$, which has a form of an exponential function of Equation (7) by SPSS.

$$f(k) = \omega \cdot k^{-\rho}. \tag{7}$$

(4) **Determination of** c: After getting $f(k)$ of Equation (7), determine a control constant for a given k by using Equations (2) and (7). That is, determine c by assigning a user-specified k to the function $f(k)$ of Equation (7).

Since $c \cdot k$ is used as the input of a k-NN query in the cc-kNN search, it should be an integer. Thus, we obtain an input integer by rounding off $c \cdot k$ to the nearest integer as in Equation (8).

$$\lfloor c \cdot k + 0.5 \rfloor = \lfloor f(k) \cdot k + 0.5 \rfloor. \tag{8}$$

[3]In general, the larger number of data we use, the more accurate function we get by the regression analysis. Thus, as the numbers n and m increase, the accuracy of the derived c-estimation function $f(k)$ also increases.

Also, if the control constant c is less than 1, the input integer of Equation (8) can be less than 1. To avoid this problem, if c is determined to be less than 1, we set it to 1, i.e., we use k as it is.

The proposed cc-kNN search changes the input of a k-NN query from k to $c \cdot k$, and thus, we need to prove its correctness. That is, we need to prove that the cc-kNN search incurs no false dismissal, i.e., it finds k data sequences correctly without any false dismissal. Theorem 4.1 shows the correctness of the cc-kNN search.

Theorem 4.1. *The cc-kNN search, which changes k to $c \cdot k$ in a k-NN query of the tolerance determination step, incurs no false dismissal and correctly finds k nearest data sequences to the query sequence.*

Proof: Based on Lemma 2.2, we use the "proof-by-contradiction" for the returned data sequences. First, let X be the k-th data sequence, which has the k-th distance to the query sequence, among k data sequences returned by the cc-kNN search, and let Y be the j-th ($j < k$) data sequence returned by the original multi-step k-NN search. Then, Equation (9) holds for the query sequence Q.

$$D_{real}(Q, Y) \leq D_{real}(Q, X). \tag{9}$$

Here we assume that the cc-kNN search does not return the sequence Y as the search result. This assumption means that Y is not chosen as a candidate sequence in a range query of the candidate filtering step. Then, the low dimensional distances from non-candidate sequences including Y to the query sequence should be larger than the tolerance $d_{\max}$ determined in the tolerance determination step. That is, Equation (10) holds.

$$D_{feature}(Q, Y) > d_{\max}. \tag{10}$$

Also, Equation (11) holds by Equation (10) and Lemma 2.1.

$$D_{real}(Q, Y) \geq D_{feature}(Q, Y) > d_{\max}. \tag{11}$$

Since $d_{\max} \geq D_{real}(Q, X)$ holds, Equation (11) is re-written as Equation (12).

$$D_{feature}(Q, Y) > D_{real}(Q, X). \tag{12}$$

However, Equation (12) is obviously false. This means that the assumption that Y is not an answer of the cc-kNN search is wrong. Therefore, by the proof-by-contradiction, Equation (9) holds, and the cc-kNN search finds all k data sequences correctly. □

5. Performance Evaluation.

5.1. Experimental data and environment.

To show the superiority of the proposed cc-kNN search, we perform an experiment using two types of time-series data. The first data set consists of 10,000 time-series of length 360, each of which is extracted from an image boundary. We call it *IMG-DATA* [13, 18]. The second data set consists of 15,000 time-series of length 60, each of which represents average daily temperatures of an arbitrary region in the world. We call it *TEMP-DATA* [21].

We first derive c-estimation functions $f(k)$ for two data sets by performing the pre-processing phase. We then measure the number of candidates and the actual execution time for the original multi-step k-NN search and the proposed cc-kNN search. The hardware platform is an HP workstation equipped with Intel Xeon 3.10GHz CPU, 4GB RAM, and 1TB HDD; its software platform is CentOS 5.9 Linux. We use C/C++ language for implementing two search methods. We use an R*-tree [3] as the multidimensional index and set its data and index page sizes to 4,096 bytes. We use PAA (piecewise approximate aggregation) [11, 24, 31] as a feature extraction function for dimensionality reduction and extract 6, 8, 12 dimensions for IMG-DATA and 5, 10, 15 dimensions for TEMP-DATA.

We use the Euclidean distance [1, 7, 17] as the similarity distance function. We randomly choose 250 sequences from each data set and use them as query sequences. Finally, we measure the number of candidates and the execution time by setting k to 1, 4, 16, 64, and 256, respectively.

5.2. **Experimental result.** Table 1 shows the results of c-estimation functions obtained from different data sets and different numbers of features. Here we set k_i to 1, 5, 10, 20, 40, 80, and 160, respectively, and use 20 as the number m of repetitions. Using the c-estimation functions of Table 1 we obtain appropriate control constants c in the cc-kNN search. Table 2 shows which control constants c we get from the c-estimation functions and which coefficients $c \cdot k$ we use for given k's in the cc-kNN search[4]. As shown in Table 2, for a small k, we get a large c; in contrast, for a large k, we get a small c. Intuitively speaking, this is obvious when we consider the number of candidates investigated in determining the tolerance. That is, for a small k, the number of candidates in the tolerance determination step is too small, and thus, we need to investigate the more number of candidates by using a large c; in contrast, for a large k, the number is already large, which means that we have already many candidates to be investigated, and thus, we use a small c.

TABLE 1. Results of c-estimation functions $f(k)$

Data set	Number of features	c-estimation function
IMG-DATA	6	$f(k) = 8.644 \cdot k^{-0.365}$
	8	$f(k) = 5.689 \cdot k^{-0.288}$
	12	$f(k) = 2.370 \cdot k^{-0.137}$
TEMP-DATA	5	$f(k) = 20.488 \cdot k^{-0.456}$
	10	$f(k) = 11.019 \cdot k^{-0.327}$
	15	$f(k) = 11.141 \cdot k^{-0.338}$

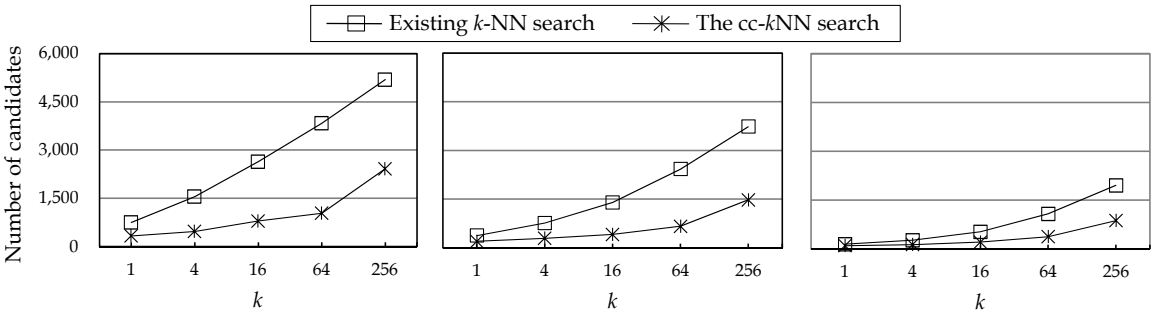
Figure 3 shows the number of candidates for IMG-DATA, where we count the candidates in the candidate filtering step for two methods, the existing multi-step k-NN search and the proposed cc-kNN search. For a k-NN query on the index, the existing solution uses k of Table 2 while the cc-kNN search uses $c \cdot k$ of Table 2. As shown in Figure 3, the cc-kNN search reduces the number of candidates by up to 72.9% compared with the existing solution. This improvement confirms that our strategy of increasing the coefficient of a k-NN query actually decreases the tolerance and accordingly reduces the number of candidates in the candidate filtering step. Figure 4 shows the number of candidates for TEMP-DATA. The result of Figure 4 is very similar to Figure 3 of IMG-DATA, and the cc-kNN search reduces the number of candidates by up to 56.1% compared with the existing solution.

Figures 5 and 6 show the actual execution time of two k-NN methods for IMG-DATA and TEMP-DATA, respectively. As shown in the figures, the proposed cc-kNN search significantly reduces the execution time compared with the existing multi-step k-NN search. In particular, the execution time in Figures 5 and 6 show very similar trends with the numbers of candidates in Figures 3 and 4. This is because the number of candidates makes a large influence on the actual execution time. In summary of Figures 5 and 6, our

[4]Actually, $\lfloor c \cdot k + 0.5 \rfloor$ is correct rather than $c \cdot k$. However, for the brief representation, we use $c \cdot k$ instead of $\lfloor c \cdot k + 0.5 \rfloor$ unless confusion occurs.

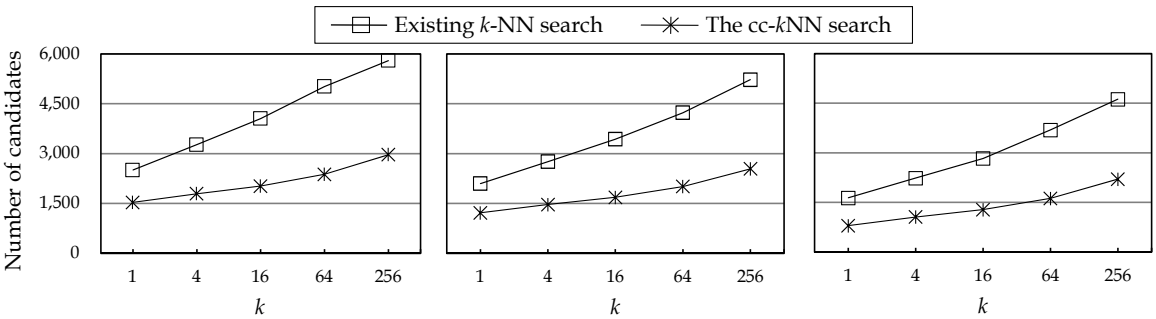
TABLE 2. Results of control constants c and coefficients $c \cdot k$

Data set	Number of features	Change of coefficients	k-NN coefficient (k)				
			1	4	16	64	256
IMG-DATA	6	c	8.64	5.21	3.14	1.89	1.14
		$c \cdot k$	9	21	50	121	292
	8	c	5.68	3.81	2.56	1.71	1.15
		$c \cdot k$	6	15	41	110	295
	12	c	2.37	1.96	1.62	1.34	1.10
		$c \cdot k$	2	8	26	86	284
TEMP-DATA	5	c	20.44	10.86	5.77	3.06	1.63
		$c \cdot k$	20	43	92	196	418
	10	c	11.01	7.00	4.45	2.85	1.79
		$c \cdot k$	11	28	71	181	460
	15	c	11.14	6.97	4.36	2.73	1.70
		$c \cdot k$	11	28	70	175	438



(a) Number of features = 6 (b) Number of features = 8 (c) Number of features = 12

FIGURE 3. Number of candidate sequences in IMG-DATA

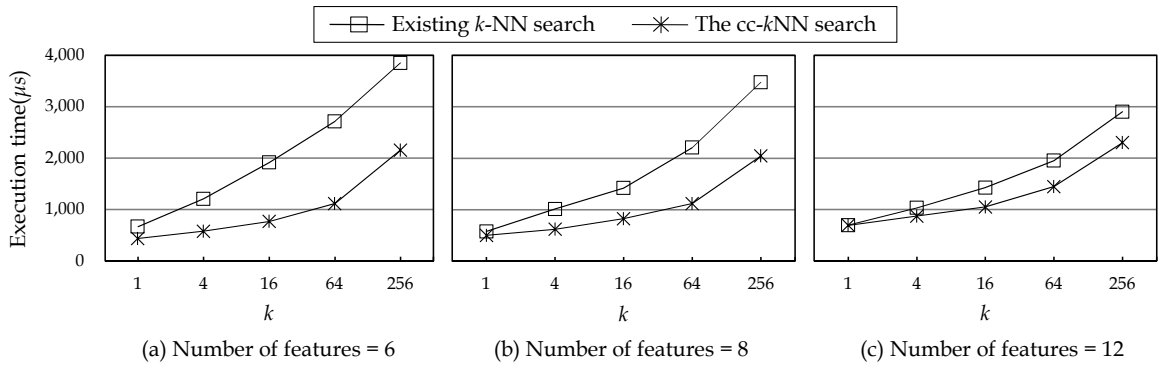


(a) Number of features = 5 (b) Number of features = 10 (c) Number of features = 15

FIGURE 4. Number of candidate sequences in TEMP-DATA

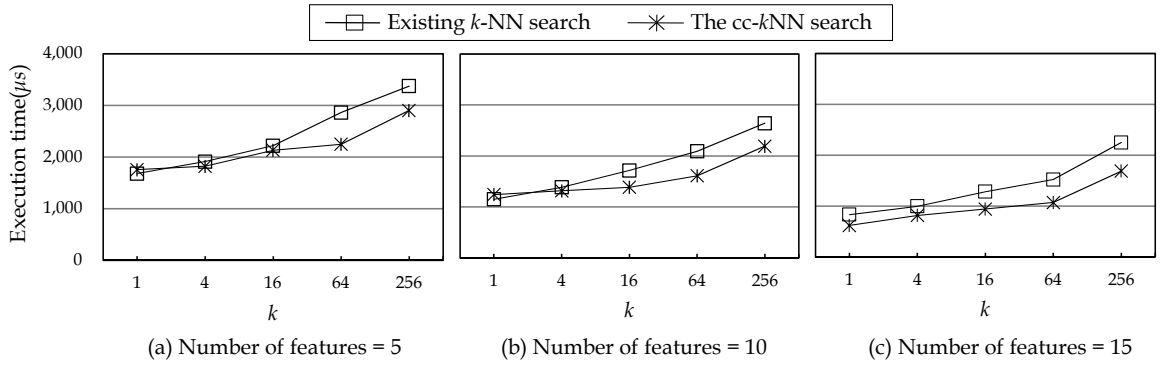
cc-kNN search reduces the execution time by up to 59.8% in IMG-DATA and 29.7% in TEMP-DATA compared with the existing solution.

Let us compare the numbers of candidates and the execution times in Figures 3 to 6. Here we note that the execution time difference of two methods is a little bit smaller than the difference of their numbers of candidates. More precisely, compared with the cc-kNN search with the existing solution, the reduction ratio of the numbers of candidates is 72.9% to 56.1%, but the reduction ratio of the execution times is merely 59.8% to 29.7%. This is because the execution time of the multi-step k-NN search (or the cc-kNN



(a) Number of features = 6 (b) Number of features = 8 (c) Number of features = 12

FIGURE 5. Execution time in IMG-DATA



(a) Number of features = 5 (b) Number of features = 10 (c) Number of features = 15

FIGURE 6. Execution time in TEMP-DATA

search) includes the time of executing the tolerance determination step, which searches the multidimensional index to find k (or $c \cdot k$) candidate sequences. Therefore, the reduction ratio of execution times is not proportional to that of the numbers of candidates in the candidate filtering step.

We also note that the experimental results are slightly different by data sets. More precisely, the cc-kNN search reduces the more number of candidates in TEMP-DATA than IMG-DATA while it reduces the more execution time in IMG-DATA than TEMP-DATA. This is because, compared with IMG-DATA of boundary images, TEMP-DATA of temperature changes has the following characteristics: (1) the average change of adjacent entries is not large, and (2) the length of sequences is small. The first observation, in which entry changes are not large, means that the difference between high and low dimensional distances is small since most dimensionality reduction functions work well in those similar entry cases, and accordingly, we can get a tight tolerance in the tolerance determination step. Thus, the cc-kNN search reduces the more number of candidates in TEMP-DATA rather than IMG-DATA. The second observation, of which the sequence is short, means that the difference between the computation time of high and low dimensional distances is not large. In TEMP-DATA, since the sequence length is relatively small, the candidate filtering step consumes a relatively little time in filtering candidates, and thus, the filtering time does not make a big influence on the total execution time. On the contrary, the long sequences of IMG-DATA incur a long time of computing high dimensional distances in the candidate filtering step, and this long filtering time makes a big influence on the total execution time. Thus, the cc-kNN search reduces more execution time in IMG-DATA rather than TEMP-DATA.

6. **Conclusions.** In this paper, we proposed an efficient multi-step k-NN search method that reduced the number of candidates and improved the overall performance. The existing multi-step k-NN search had a problem of incurring a large number of candidates in a range query due to information loss of dimensionality reduction. Based on the observation, we proposed the coefficient control multi-step k-NN search, called *cc-kNN search*, which produced a tight tolerance for a range query and reduced the number of candidates. The proposed cc-kNN search used a simple but efficient intuition that the use of $c \cdot k$ rather than k in a k-NN query on the index would produce a tight tolerance for the next range query. For this, we first defined a control constant c for determining the increment of k, and we then derived a c-estimation function $f(k)$ and k_i-time functions $g_i(x)$ by a preliminary experiment of the pre-processing phase. We formally derived those functions by using the regression analysis and used them for determining an appropriate c for a given k. We also formally proved the correctness of the proposed cc-kNN search by presenting Theorem 4.1. Finally, we empirically showcased the superiority of the cc-kNN search by varying data sets, numbers of dimensions, and coefficients k. Based on these results, we believe that our cc-kNN search is an excellent solution that improves the performance of the existing multi-step k-NN search without change of its basic structure and intrinsic working mechanism.

As the future research directions, we suggest two important issues as follows. First, we will try to apply the cc-kNN search to a few real applications of multimedia information retrieval [8, 12, 29] to improve their performance in large-scale multimedia databases. Second, we will also try to modify the indexing mechanism of the cc-kNN search by exploiting the LSH-base B-tree [28] to achieve much higher performance.

Acknowledgement. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and future Planning (NRF-2014R1A2A2A01002548).

REFERENCES

[1] R. Agrawal, C. Faloutsos and A. Swami, Efficient similarity search in sequence databases, *Proc. of the 4th Int'l Conf. on Foundations of Data Organization and Algorithms*, Chicago, Illinois, pp.69-84, 1993.

[2] L. Akoglu, R. Khandekar, V. Kumar, S. Parthasarathy, D. Rajan and K.-L. Wu, Fast nearest neighbor search on large time-evolving graphs, *Proc. of the European Conf. on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, Nancy, France, pp.17-33, 2014.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider and B. Seeger, The R*-tree: An efficient and robust access method for points and rectangles, *Proc. of Int'l. Conf. on Management of Data*, Atlantic City, New Jersey, pp.322-331, 1990.

[4] S. Berchtold, C. Bohm and H.-P. Kriegel, The pyramid-technique: Towards breaking the curse of dimensionality, *Proc. of Int'l Conf. on Management of Data*, Seattle, Washington, pp.142-153, 1998.

[5] R. L. Burden and J. D. Faires, *Numerical Analysis*, 9th Edition, Brooks/Cole, 2010.

[6] S. C. Chapra, *Numerical Methods for Engineers*, 6th Edition, McGraw-Hill Science, 2010.

[7] K.-P. Chan, A. W.-C. Fu and C. T. Yu, Haar wavelets for efficient similarity search of time-series: With and without time warping, *IEEE Trans. Knowledge and Data Engineering*, vol.15, no.3, pp.686-705, 2003.

[8] T. N. Chan, M. L. Yiu and K. A. Hua, A progressive approach for similarity search on matrix, *Proc. of the 14th Int'l Symp. on Spatial and Temporal Databases*, Hong Kong, China, pp.373-390, 2015.

[9] C. Faloutsos, M. Ranganathan and Y. Manolopoulos, Fast subsequence matching in time-series databases, *Proc. of Int'l Conf. on Management of Data*, Minneapolis, Minnesota, pp.419-429, 1994.

[10] T.-C. Fu, A review on time series data mining, *Engineering Applications of Artificial Intelligence*, vol.24, no.1, pp.164-181, 2011.

[11] W.-S. Han, J. Lee, Y.-S. Moon and H. Jiang, Ranked subsequence matching in time-series databases, *Proc. of the 33rd Int'l. Conf. on Very Large Data Bases*, Vienna, Austria, pp.423-434, 2007.

[12] M. Kim, K.-Y. Whang and Y.-S. Moon, Horizontal reduction: Instance-level dimensionality reduction for similarity search in large document databases, *Proc. of the 28th Int'l. Conf. on Data Engineering*, Washington D.C., pp.1061-1072, 2012.
[13] B.-S. Kim, Y.-S. Moon, M.-J. Choi and J. Kim, Interactive noise-controlled boundary image matching using the time-series moving average transform, *Multimedia Tools and Applications*, vol.72, no.3, pp.2543-2571, 2014.
[14] F. Korn, N. Sidiropoulos, C. Faloutsos, E. Siegel and Z. Protopapas, Fast nearest neighbor search in medical image databases, *Proc. of the 22nd Int'l. Conf. on Very Large Data Bases*, Bombay, India, pp.215-226, 1996.
[15] Y.-S. Moon, K.-Y. Whang and W.-K. Loh, Duality-based subsequence matching in time-series databases, *Proc. of the 17th Int'l. Conf. on Data Engineering*, Heidelberg, Germany, pp.263-272, 2001.
[16] Y.-S. Moon, K.-Y. Whang and W.-S. Han, General match: A subsequence matching method in time-series databases based on generalized windows, *Proc. of Int'l. Conf. on Management of Data*, Madison, Wisconsin, pp.382-393, 2002.
[17] Y.-S. Moon and J. Kim, Efficient moving average transform-based subsequence matching algorithms in time-series databases, *Information Sciences*, vol.177, no.23, pp.5415-5431, 2007.
[18] Y.-S. Moon, B.-S. Kim, M. S. Kim and K.-Y. Whang, Scaling-invariant boundary image matching using time-series matching techniques, *Data and Knowledge Engineering*, vol.69, no.10, pp.1022-1042, 2010.
[19] Y.-S. Moon and B. S. Lee, Safe MBR-transformation in similar sequence matching, *Information Sciences*, vol.270, pp.28-40, 2014.
[20] Y.-S. Moon and W.-K. Loh, Triangular inequality-based rotation-invariant boundary image matching for smart devices, *Multimedia Systems*, vol.21, no.1, pp.15-28, 2015.
[21] *National Climatic Data Center*, http://www.ncdc.noaa.gov.
[22] M. O'Mahony, N. Hurley, N. Kushmerick and G. Silvestre, Collaborative recommendation: A robustness analysis, *ACM Trans. Internet Technology*, vol.4, no.4, pp.344-377, 2004.
[23] F. Rasheed, M. Al-Shalalfa and R. Alhajj, Efficient periodicity mining in time series databases using suffix trees, *IEEE Trans. Knowledge and Data Engineering*, vol.23, no.1, pp.79-94, 2011.
[24] G. Roh, J. Roh, S. Hwang and B. Yi, Supporting pattern matching queries over trajectories on road networks, *IEEE Trans. Knowledge and Data Engineering*, vol.23, no.11, pp.1753-1758, 2011.
[25] B. G. Samuel and J. S. Neil, *Using SPSS for Windows and Macintosh: Analyzing and Understanding Data*, 6th Edition, Pearson College Div, 2010.
[26] T. Seidl and H. P. Kriegel, Optimal multi-step k-nearest neighbor search, *Proc. of Int'l. Conf. on Management of Data*, Seattle, Washington, pp.154-165, 1998.
[27] T. Seidl, *Adaptable Similarity Search in 3-D Spatial Database Systems*, Herbert Utz Verlag, 1998.
[28] Y. Tao, K. Yi, C. Sheng and P. Kalnis, Quality and efficiency in high dimensional nearest neighbor search, *Proc. of Int'l Conf. on Management of Data*, Providence, Rhode Island, USA, pp.563-575, 2009.
[29] M. S. Uysal, C. Beecks, D. Sabinasz and T. Seidl, Large-scale efficient and effective video similarity search, *Proc. of the 2015 Workshop on Large-Scale and Distributed System for Information Retrieval*, Melbourne, Australia, pp.3-8, 2015.
[30] J. Zhou and A. Tung, SMiLer: A semi-lazy time series prediction system for Sensors, *Prof. of Int'l Conf. on Management of Data*, Melbourne, Australia, pp.1871-1886, 2015.
[31] Y. Zhu and D. Shasha, Warping indexes with envelope transforms for query by humming, *Proc. of Int'l. Conf. on Management of Data*, San Diego, California, pp.181-192, 2003.