International Journal of Innovative Computing, Information and Control
Volume **9**, Number **4**, April **2013**

ICIC International ©2013 ISSN 1349-4198
pp. **1423–1438**

LOWER-DIMENSIONAL TRANSFORMATIONS FOR HIGH-DIMENSIONAL MINIMUM BOUNDING RECTANGLES

Mi-Jung Choi[1], Woong-Kee Loh[2] and Yang-Sae Moon[1,*]

[1]Department of Computer Science
Kangwon National University
192-1 Hyoja2-Dong, Chunchon, Kangwon 200-701, Republic of Korea
mjchoi@kangwon.ac.kr; *Corresponding author: ysmoon@kangwon.ac.kr

[2]Department of Multimedia
Sungkyul University
400-10 Anyang8-Dong, Anyang, Gyeonggi 430-742, Republic of Korea
woong@sungkyul.ac.kr

Received January 2012; revised May 2012

Abstract. *There have been many research efforts on the lower-dimensional transformation of high-dimensional points. However, a lot of real data such as time-series sequences and streaming data can be modeled as an MBR (Minimum Bounding Rectangle) rather than a point in a high-dimensional space. To store and search those high-dimensional MBRs in a multidimensional index, we need to transform a high-dimensional MBR itself to a low-dimensional MBR directly. To tackle the problem, we first present a new notion of lower-dimensional MBR transformation. The lower-dimensional MBR transformation should construct a low-dimensional MBR by containing all the low-dimensional points to which an infinite number of high-dimensional points in the MBR are transformed. As the naive solutions for DFT (discrete Fourier transform) and DCT (discrete Cosine transform), we then propose DFTnaive and DCTnaive, respectively, which use all the vertex points on a high-dimensional MBR. These naive solutions, however, require a huge number of transformations as the dimension increases. To solve this problem, we further propose DFTadv and DCTadv, which use only two points (a lower-left and an upper-right points) rather than all the vertex points on an MBR. By presenting related theorems, we also formally prove that all the proposed solutions perform the lower-dimensional MBR transformation correctly. Experimental results show that our advanced solution outperforms the naive solution by up to 13,100 times, when the dimension is 16, and the improvement becomes larger as the dimension increases. These results indicate that the proposed notion of lower-dimensional MBR transformation provides a very practical framework for a variety of time-series data applications.*

Keywords: Lower-dimensional MBR transformation, High-dimensional MBR, Discrete Fourier transform (DFT), Discrete Cosine transform (DCT)

1. Introduction.

Time-series data are sequences of real numbers representing values at specific time points – examples are stock prices, exchange rates, weather data, financial data, network traffic data, etc. Finding data sequences similar to a given query sequence from the database is called *similar sequence matching* or *time-series matching* [1, 7, 22]. Time-series matching has been widely used in many practical applications including image matching, handwritten recognition, speech recognition, query by humming, privacy-preserving data publication, and biological sequence matching [10, 13, 16, 20, 26, 27]. Our solution can be used for such practical applications as it can improve the overall performance of time-series matching.

A *lower-dimensional transformation* is defined as an operation of transforming a high-dimensional point to a low-dimensional point [1, 21]. This lower-dimensional transformation is widely used in many applications such as data mining [11], pattern recognition [29], and GIS (Geographic Information System) [9]. It transforms a high-dimensional point with hundreds or thousands of dimensions to a low-dimensional point with one to six dimensions. Storing high-dimensional points in a multimensional index [3] causes the high dimensionality problem (called *dimensionality curse*) [4] and requires excessive index space. To avoid these problems, we generally use the lower-dimensional transformation to reduce dimensionality of points to be stored in the index [7, 24].

In this paper we tackle the problem of transforming a high-dimensional *MBR* (Minimum Bounding Rectangle) to a low-dimensional MBR. An n-dimensional MBR is defined as a minimum-sized hyper-rectangle that contains an n-dimensional object or multiple n-dimensional points [2, 3, 8], and it is represented as a *lower-left point* and an *upper-right point*. There have been many research efforts on the lower-dimensional transformation of a high-dimensional point, but to our knowledge there has been no research result on that of a high-dimensional MBR itself. However, a lot of real data such as time-series sequences and streaming data can be modeled as an MBR rather than a point in a high-dimensional space. The followings are representative motivating examples of practical use of such high-dimensional MBRs.

- The highest and lowest prices (or the bid and ask prices) of a stock item can be represented as a high-dimensional sequence with upper and lower values, and the sequence can be modeled as a high-dimensional MBR. We often want to find some stock items showing a similar trend with one's own interesting stock item, which is also modeled as a high-dimensional MBR. Searching and grouping stock items having similar trends have been important tasks in stock data analysis [19, 30].
- Temperatures can be represented as a high-dimensional sequence with the highest and lowest values and also be modeled as a high-dimensional MBR. For various temperature data from different seasons or different regions, we may find similar temperature clusters to group seasons or regions in a meaningful way.
- In rotation-invariant boundary image matching [17, 25], image boundaries can be modeled as high-dimensional MBRs. We construct a high-dimensional MBR for a query image and search its similar images from the large image database. In this process, we can use the lower-dimensional MBR transformation to get a low-dimensional query MBR and search on the index.
- High-dimensional MBRs are also used in subsequence matching [7, 22], where we construct an MBR by bounding many high-dimensional subsequences. As in [23], if we have a lower-dimensional MBR transformation, we can build an index fast or perform subsequence matching efficiently.

As you can see examples above, high-dimensional MBRs are widely used in many time-series applications. We here note that, like high-dimensional points, storing high-dimensional MBRs in an index is also difficult [4], and thus, we need a systematic way that transforms a high-dimensional MBR to a low-dimensional MBR. Hereafter, we abbreviate "high-dimensional" to *HD* and "low-dimensional" to *LD*.

Transforming an HD MBR to an LD MBR can be done by the following simple way: 1) every possible HD point in the HD MBR is transformed to an LD point, and 2) all the transformed LD points are included in the LD MBR. This simple method, however, is not practically applicable since the number of HD points in an MBR is infinite. To solve this problem, we consider a finite number of points for the lower-dimensional transformation of an HD MBR. Various transforms including DFT (Discrete Fourier Transform), DCT

(Discrete Cosine Transform), and Wavelet transform are used as the lower-dimensional transformation for HD points. Among these transforms, DFT and DCT are widely used in similar sequence matching [1, 12, 25] and multimedia data retrieval [25, 31]. Thus, in this paper we provide DFT-based and DCT-based lower-dimensional transformations of HD MBRs.

We first present the notion of *lower-dimensional MBR transformation* that transforms an HD MBR itself to an LD MBR directly. We then propose two DFT-based lower-dimensional MBR transformations. The first one is a naive solution, called *DFTnaive*, which transforms every vertex point on an HD MBR to an LD point using DFT and constructs an LD MBR by bounding all the transformed LD points. We formally prove that DFTnaive performs the lower-dimensional MBR transformation correctly. However, DFTnaive has a critical problem that it requires a huge number of lower-dimensional transformations since the number is proportional to $O(2^n)$, where n is the dimension of an HD MBR, which is too large and practically useless. Thus, as the second solution, we propose *DFTadv*, an advanced version of DFTnaive. DFTadv uses only two points (a lower-left and an upper-right points) rather than all vertex points on an MBR. We also formally prove correctness of DFTadv. Through analysis, we show that the number of transformations required in DFTadv is only $O(1)$, which is drastically reduced from $O(2^n)$ of DFTnaive. After then, we propose two DCT-based lower-dimensional MBR transformations. Like DFT-based solutions, we first present *DCTnaive* that uses all vertex points on an MBR, and then explain *DCTadv* that uses only a lower-left and an upper-right points on an MBR.

Through analysis and experiments, we show superiority of the proposed DFTadv and DCTadv. By deriving the numbers of transformations required in DFTnaive and DFTadv and those in DCTnaive and DCTadv, we analytically explain significant superiority of DF-Tadv and DCTadv. According to the analysis, DFTnaive and DCTnaive with exponential complexity are practically useless, while DFTadv and DCTadv with constant complexity are very practically useful for time-series matching applications. Moreover, experimental results on real stock data show that, compared with DFTnaive and DCTnaive, DFTadv and DCTadv significantly reduce the execution time by up to 13,100 and 11,600 times, respectively, when the dimension is 16. More importantly, the performance improvement becomes larger as the dimension increases.

The rest of this paper is organized as follows. Section 2 describes existing work related to lower-dimensional transformations. Section 3 defines the lower-dimensional MBR transformation. Section 4 proposes DFT-based and DCT-based lower-dimensional MBR transformations. Section 5 presents the results of performance evaluation. Section 6 summarizes and concludes the paper.

2. **Related Work.**

Various transforms including DFT, DCT, and Wavelet transform are used as the lower-dimensional transformation of HD points. DFT is most widely used in many applications, especially in similar sequence matching [1, 7, 12, 20]. DCT is used in applications on multimedia data [25, 31] or stream data [14]. Wavelet transform is also used as the lower-dimensional transformation in [5, 27]. Besides these transforms, PAA (Piecewise Aggregate Approximation) [15] and SVD (Singluar Value Decomposition) [18] were introduced as the lower-dimensional transformation. All these transformations, however, focused on transforming HD points (sequences) or HD images to LD ones, and they cannot be directly applied to the lower-dimensional MBR transformation.

In this paper, we handle the lower-dimensional MBR transformation for DFT and DCT. Table 1 summarizes the notation to be used throughout the paper. Based on the notation in Table 1, we first present the definition of DFT. DFT transforms an n-dimensional

Table 1. Summary of notation

Symbols	Definitions
X	An HD point. $(= \{x_0, x_1, \ldots, x_{n-1}\})$.
X^T	An LD point transformed from the HD point X by the transformation T. $(= \{x_0^T, x_1^T, \ldots, x_{m-1}^T\})$
$[L, U]$	An HD MBR of which lower-left and upper-right points are L and U, respectively. $(= [\{l_0, l_1, \ldots, l_{n-1}\}, \{u_0, u_1, \ldots, u_{n-1}\}])$
$[L, U]^T$ $= [\Lambda, \Upsilon]$	An LD MBR transformed from the HD MBR $[L, U]$ by the transformation T. $(= [\{\lambda_0, \lambda_1, \ldots, \lambda_{n-1}\}, \{\upsilon_0, \upsilon_1, \ldots, \upsilon_{n-1}\}])$
$X \in [L, U]$	The point X is contained in the MBR $[L, U]$. (i.e., for every i, $l_i \leq x_i \leq u_i$)

point X to a new n-dimensional point Y $(= \{y_0, y_1, \ldots, y_{n-1}\})$ in a complex number space, where each y_i is defined as the following Equation (1) [28]:

$$y_i = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t \cos(-2\pi i t/n) + \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t \sin(-2\pi i t/n) \cdot j, \quad 0 \leq i \leq n-1. \tag{1}$$

DFT concentrates most of the energy into the first few coefficients, and thus only a few coefficients extracted from the transformed point Y are used for the lower-dimensional transformation [1, 7]. Definition 2.1 shows the traditional lower-dimensional transformation using DFT.

Definition 2.1. *The DFT-based lower-dimensional transformation is defined as an operation of transforming an n-dimensional point X to an $m(\ll n)$-dimensional point X^{DFT} of $\{x_0^{DFT}, x_1^{DFT}, \ldots, x_{m-1}^{DFT}\}$, where each x_i^{DFT} is obtained by Equation (2). In Equation (2), $0 \leq i \leq m-1$.*

$$x_i^{DFT} = \begin{cases} \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t \cos(-2\pi \lfloor i/2 \rfloor t/n), & \text{if } i \text{ is even;} \\ \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t \sin(-2\pi \lfloor i/2 \rfloor t/n), & \text{if } i \text{ is odd.} \end{cases} \tag{2}$$

DCT-based lower-dimensional transformation also uses the first few coefficients of an n-dimensional point Y to which an n-dimensional point X is transformed by DCT [5]. Definition 2.2 shows the traditional lower-dimensional transformation using DCT [28].

Definition 2.2. *The DCT-based lower-dimensional transformation is defined as an operation of transforming an n-dimensional point X to an $m(\ll n)$-dimensional point X^{DCT} of $\{x_0^{DCT}, x_1^{DCT}, \ldots, x_{m-1}^{DCT}\}$, where each x_i^{DCT} is obtained by Equation (3). In Equation (3), $0 \leq i \leq m-1$.*

$$x_i^{DCT} = \frac{2 \cdot c(i)}{n} \sum_{t=0}^{n-1} x_t \cos\left(\frac{(2t+1)i\pi}{2n}\right), \text{ where } c(i) = \begin{cases} \sqrt{2}/2, & \text{if } i = 0; \\ 1, & \text{if } i = 1, 2, \ldots, m-1. \end{cases} \tag{3}$$

In the previous work [23], we have presented a concept of MBR-Safe transformation and proposed *mbrDFT* as its example implementation. In subsequence matching [7, 22], we need to construct HD MBRs bounding multiple subsequences and convert those HD MBRs to LD MBRs for a multidimensional index. In [23], we have used mbrDFT for subsequence matching to improve the performance of obtaining query or data MBRs. The previous work, however, focused on only subsequence matching with DFT. Thus, we can say that the lower-dimensional MBR transformation to be proposed is a generalized version of [23] since it can be used for any MBR-based time-series applications, and it also handles both DCT as well as DFT. To show the generality of the proposed MBR transformation, we perform the experiments on subsequence matching in Section 5.

We introduce two recent applications of time-series matching to which we can apply lower-dimensional MBR transformations. First, Choi et al. [6] have presented a novel approach that published time-series data by preserving their distance orders as well as their privacy. They have tried to get mining results with relatively higher accuracy while not disclosing privacy of individual time-series. Similar to HD points [6, 26], we may need to preserve privacy of HD MBRs, and in this case, we can use Choi et al.'s research result to publish HD MBRs with preserving privacy and distance orders. Second, we can use the MBR transformation to boundary image matching [17, 25], where we have solved the image matching problems in the time-series domain rather than the image domain. As we mentioned in Section 1, in rotation-invariant boundary image matching [17], we can model image boundaries as HD MBRs. More precisely, we construct HD MBRs for query or data images, store data MBRs in an index, and identify data images similar to the query MBRs using the index. In this process, we can use the lower-dimensional MBR transformation to get LD MBRs from HD MBRs for query or data image time-series.

3. Problem Definition.

The lower-dimensional MBR transformation is an operation of constructing an LD MBR by containing every LD point to which every possible HD point in the HD MBR is transformed. Here, the number of possible HD points contained in the HD MBR is infinite. We formally present the concept of lower-dimensional MBR transformation and its correctness as follows:

Definition 3.1. *For an n-dimensional point X and an n-dimensional MBR $[L, U]$, if a transformation T satisfies the following Equation (4), then we say T performs the lower-dimensional MBR transformation correctly.*

$$X \in [L, U] \Rightarrow X^T \in [L, U]^T \tag{4}$$

Definition 3.1 means that, to guarantee correctness of the lower-dimensional MBR transformation, an LD MBR $[L, U]^T$ to which an HD MBR $[L, U]$ is transformed should contain every LD point X^T to which every possible X in $[L, U]$ is transformed.

Figure 1 shows the traditional lower-dimensional transformation of an HD point and the proposed lower-dimensional MBR transformation. As shown in the figure, the traditional transformation maps a point in an HD space to a point in an LD space; in contrast, the lower-dimensional MBR transformation maps an MBR in an HD space to an MBR in an LD space. Next, Figure 2 shows an example of a correct lower-dimensional MBR transformation. As depicted in the figure, if an arbitrary point X is contained in $[L, U]$ (i.e., $X \in [L, U]$), the transformed point X^T is also contained in the transformed MBR $[L, U]^T$ (i.e., $X^T \in [L, U]^T$). Thus, we can say T performs the lower-dimensional MBR transformation correctly.

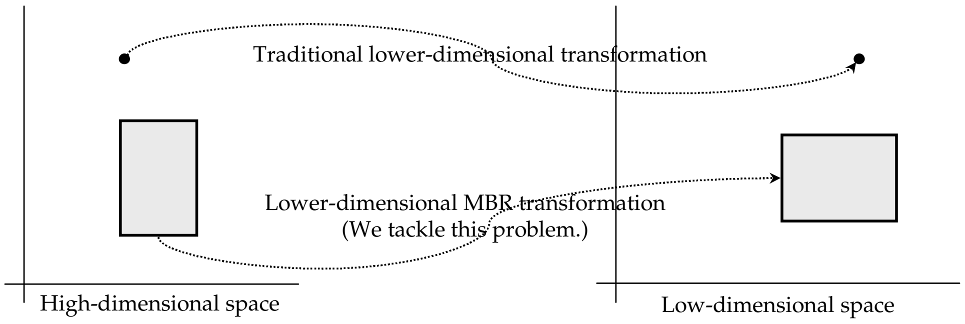

Traditional lower-dimensional transformation
Lower-dimensional MBR transformation
(We tackle this problem.)
High-dimensional space
Low-dimensional space

FIGURE 1. Traditional lower-dimensional point and MBR transformations

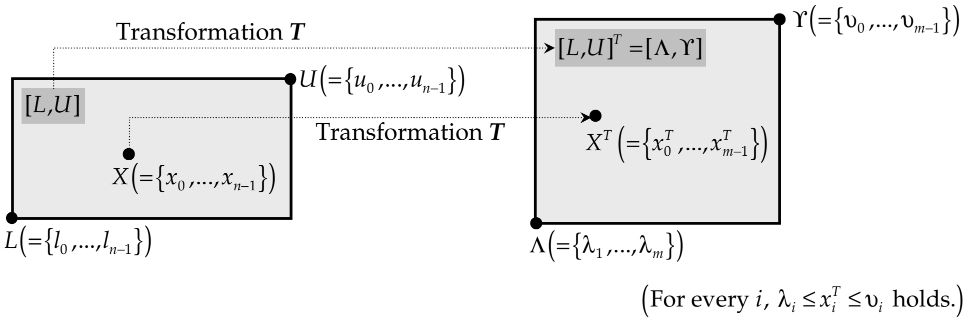

Transformation T
$[L,U]^T=[\Lambda,\Upsilon]$
$\Upsilon(=\{\upsilon_0,\dots,\upsilon_{m-1}\})$
$U(=\{u_0,\dots,u_{n-1}\})$
$[L,U]$
Transformation T
$X^T(=\{x_0^T,\dots,x_{m-1}^T\})$
$X(=\{x_0,\dots,x_{n-1}\})$
$L(=\{l_0,\dots,l_{n-1}\})$
$\Lambda(=\{\lambda_1,\dots,\lambda_m\})$
(For every i, $\lambda_i \le x_i^T \le \upsilon_i$ holds.)

FIGURE 2. An example of a correct lower-dimensional MBR transformation

4. **Lower-Dimensional MBR Transformations.** In this section we propose lower-dimensional MBR transformations for the most widely used DFT and DCT, respectively. Section 4.1 proposes DFTnaive and DFTadv by extending the traditional DFT to the lower-dimensional MBR transformation. Section 4.2 also proposes DCTnaive and DCTadv by extending DCT in a similar way.

4.1. **DFT-based lower-dimensional MBR transformations.** As we mentioned in Section 1, the simple transformation method, which constructs an LD MBR by transforming an infinite number of HD points in an HD MBR and by containing all the transformed LD points, is not practically applicable. Thus, we first present a DFT-based naive approach that uses a finite number of vertex points on an HD MBR. Definition 4.1 shows a formal definition of the DFT-based naive approach.

Definition 4.1. *For an n-dimensional MBR $[L,U]$, DFTnaive is defined as an operation that constructs an $m(\ll n)$-dimensional MBR by performing DFT-based lower-dimensional transformations on all 2^n vertex points on $[L,U]$.*

Intuitively speaking, since the vertex points on an MBR are representatives of all the possible points contained in the MBR, DFTnaive that considers all the vertex points will perform the lower-dimensional MBR transformation correctly. Theorem 4.1 guarantees correctness of DFTnaive.

Theorem 4.1. *For an n-dimensional point X and an n-dimensional MBR $[L,U]$, DFTnaive satisfies the equation of $X \in [L,U] \Rightarrow X^{DFT} \in [L,U]^{DFTnaive}$. That is, DFTnaive performs the lower-dimensional MBR transformation correctly.*

Proof: Since $X \in [L,U]$, $l_t \le x_t \le u_t$ holds for every t $(0 \le t \le n-1)$. We now proceed the proof by two cases: 1) the first case where i in x_i^{DFT} is even, and 2) the second one where i is odd.
1) The case where i in x_i^{DFT} of X^{DFT} is even: Among vertex points on the MBR $[L,U]$, there exist two vertex points P and Q that are constructed by the following Equation (5). In Equation (5), $\theta = -2\pi\lfloor i/2\rfloor t/n$.

$$p_t = \begin{cases} l_t, & \text{if } \cos\theta \ge 0; \\ u_t, & \text{if } \cos\theta < 0; \end{cases},\quad q_t = \begin{cases} u_t, & \text{if } \cos\theta \ge 0; \\ l_t, & \text{if } \cos\theta < 0; \end{cases},\ \text{where } 0 \le t \le n-1. \tag{5}$$

Then, $p_t^{DFT} \le x_t^{DFT} \le q_t^{DFT}$ holds by Equation (2) in Definition 2.1. The detailed reason is as follows. If $\cos\theta$ is positive, $l_t\cos\theta \le x_t\cos\theta \le u_t\cos\theta$ holds since $l_t \le x_t \le u_t$. Similarly, if $\cos\theta$ is negative, $u_t\cos\theta \le x_t\cos\theta \le l_t\cos\theta$ holds. Thus, $\sum_{t=0}^{n-1} p_t\cos\theta$, which is obtained by adding $l_t\cos\theta$ if $\cos\theta$ is positive and $u_t\cos\theta$ if $\cos\theta$ is negative, is less than or equal to $\sum_{t=0}^{n-1} x_t\cos\theta$. It means that $\frac{1}{\sqrt{n}}\sum_{t=0}^{n-1} p_t\cos\theta\ (= p_i^{DFT})$ is less than or equal to

$\frac{1}{\sqrt{n}}\sum_{t=0}^{n-1} x_t\cos\theta$ $(= x_i^{DFT})$. Analogously, $\sum_{t=0}^{n-1} q_t\cos\theta$, which is obtained by adding $u_t\cos\theta$ if $\cos\theta$ is positive and $l_t\cos\theta$ if $\cos\theta$ is negative, is greater than or equal to $\sum_{t=0}^{n-1} x_t\cos\theta$. Thus, $\frac{1}{\sqrt{n}}\sum_{t=0}^{n-1} q_t\cos\theta$ $(= q_i^{DFT})$ is greater than or equal to $\frac{1}{\sqrt{n}}\sum_{t=0}^{n-1} x_t\cos\theta$ $(= x_i^{DFT})$. Therefore, $p_i^{DFT} \leq x_i^{DFT} \leq q_i^{DFT}$ holds for every case where i in x_i^{DFT} is even.
2) The case where i in x_i^{DFT} of X^{DFT} is odd: Among vertex points on the MBR $[L, U]$, there exist two vertex points R and S that are constructed by the following Equation (6). In Equation (6), $\theta = -2\pi\lfloor i/2\rfloor t/n$.

$$r_t = \begin{cases} l_t, & \text{if } \sin\theta \geq 0; \\ u_t, & \text{if } \sin\theta < 0; \end{cases}, \quad s_t = \begin{cases} u_t, & \text{if } \sin\theta \geq 0; \\ l_t, & \text{if } \sin\theta < 0. \end{cases}, \text{ where } 0 \leq t \leq n-1. \tag{6}$$

Then, $r_t^{DFT} \leq x_t^{DFT} \leq s_t^{DFT}$ also holds by the similar steps described in the case 1) above. Here, since P^{DFT}, Q^{DFT}, R^{DFT} and S^{DFT} are contained in the LD MBR $[L, U]^{DFTnaive}$ by Definition 4.1, X^{DFT} is also contained in $[L, U]^{DFTnaive}$, i.e., $X^{DFT} \in [L, U]^{DFTnaive}$ is satisfied. Therefore, the equation of $X \in [L, U] \Rightarrow X^{DFT} \in [L, U]^{DFTnaive}$ holds. Also, we note that the left-point of $[L, U]^{DFTnaive}$ can be obtained from p_i^{DFT} (i is even) and r_i^{DFT} (i is odd), and the right-point from q_i^{DFT} (i is even) and s_i^{DFT} (i is odd). □

DFTnaive, however, has a critical problem that it requires a huge number of transformations as the dimension increases. For an n-dimensional MBR, DFTnaive requires 2^n DFT-based lower-dimensional transformations since the number of vertex points on the n-dimensional MBR is 2^n. Thus, DFTnaive cannot be practically used for HD MBRs with tens to thousands of dimensions. To solve this problem, we next propose DFTadv that drastically reduces the number of transformations compared with DFTnaive. Shortly speaking, DFTadv considers P, Q, R and S only, which are introduced in the proof steps of Theorem 4.1. It means that DFTadv uses a lower-left and an upper-right points rather than all vertex points by extending the traditional definition of DFT to the lower-dimensional MBR transformation. Definition 4.2 presents a formal definition of DFTadv.

Definition 4.2. *For an n-dimensional MBR $[L, U]$, DFTadv is defined as an operation that constructs an $m(\ll n)$-dimensional MBR of which lower-left and upper-right points are Λ and Υ, respectively, in Equation (7). In Equation (7), $\theta = -2\pi\lfloor i/2\rfloor t/n$ and $0 \leq i \leq m-1$.*

$$\lambda_i = \begin{cases} \frac{1}{\sqrt{n}}\sum_{t=0}^{n-1} a_t\cos\theta, & \text{if } i \text{ is even;} \\ \frac{1}{\sqrt{n}}\sum_{t=0}^{n-1} b_t\sin\theta, & \text{if } i \text{ is odd.} \end{cases}, \quad \upsilon_i = \begin{cases} \frac{1}{\sqrt{n}}\sum_{t=0}^{n-1} c_t\cos\theta, & \text{if } i \text{ is even;} \\ \frac{1}{\sqrt{n}}\sum_{t=0}^{n-1} d_t\sin\theta, & \text{if } i \text{ is odd.} \end{cases},$$

$$\text{where } \begin{cases} a_t = l_t, c_t = u_t, & \text{if } \cos\theta \geq 0; \\ a_t = u_t, c_t = l_t, & \text{if } \cos\theta < 0; \\ b_t = l_t, d_t = u_t, & \text{if } \sin\theta \geq 0; \\ b_t = u_t, d_t = l_t, & \text{if } \sin\theta < 0. \end{cases} \tag{7}$$

DFTadv in Definition 4.2 is derived from DFTnaive to reduce the number of transformations. To perform the lower-dimensional MBR transformation correctly, DFTadv makes Λ and Υ of the transformed MBR $[\Lambda, \Upsilon]$ contain every possible point that can be generated from the original MBR by the DFT-based lower-dimensional transformation. Theorem 4.2 shows correctness of DFTadv.

Theorem 4.2. *For an n-dimensional point X and an n-dimensional MBR $[L, U]$, DFTadv satisfies the equation of $X \in [L, U] \Rightarrow X^{DFT} \in [L, U]^{DFTadv}$. That is, DFTadv performs the lower-dimensional MBR transformation correctly.*

Proof: In the proof steps of Theorem 4.1, we derived p_i^{DFT}, q_i^{DFT}, r_i^{DFT} and s_i^{DFT}. We then note that Equation (7) of DFTadv simply denotes p_i^{DFT} or r_i^{DFT} as λ_i and q_i^{DFT} or s_i^{DFT} as υ_i. Thus, the MBR of DFTadv is the same as that of DFTnaive. It is trivial because p_i^{DFT} or r_i^{DFT} and q_i^{DFT} or s_i^{DFT} are a lower-left and an upper-right points in DFTnaive, and λ_i $(= p_i^{DFT}$ or $r_i^{DFT})$ and υ_i $(= q_i^{DFT}$ or $s_i^{DFT})$ are those in DFTadv. Therefore, DFTadv also performs the lower-dimensional MBR transformation correctly since we have already shown correctness of DFTnaive in Theorem 4.1. □

Example 4.1 shows an example that DFTnaive and DFTadv perform the lower-dimensional MBR transformation.

Example 4.1. *We want to transform a 4-dimensional MBR* $[L, U]$ *to a 2-dimensional MBR where* $L = \{2.0, 3.0, 4.0, 5.0\}$ *and* $U = \{4.0, 5.0, 6.0, 7.0\}$. *According to Definition 4.1, DFTnaive performs total 16 DFT-based lower-dimensional transformations for 16 4-dimensional vertex points,* $\{2.0, 3.0, 4.0, 5.0\}$, $\{2.0, 3.0, 4.0, 7.0\}$, $\{2.0, 3.0, 6.0, 5.0\}$, *. . . ,* $\{4.0, 5.0, 6.0, 5.0\}$ *and* $\{4.0, 5.0, 6.0, 7.0\}$. *Thus, it produces the corresponding 16 2-dimensional points,* $\{7.0, -1.0\}$, $\{8.0, -1.0\}$, $\{8.0, -2.0\}$, *. . . ,* $\{10.0, -1.0\}$ *and* $\{11.0, -1.0\}$. *After then, DFTnaive constructs a 2-dimensional MBR* $[\{7.0, -2.0\}, \{11.0, 0.0\}]$ *by containing all the 2-dimensional points. On the other hand, according to Definition 4.2, DFTadv obtains a 2-dimensional MBR* $[\Lambda, \Upsilon]$ *directly by computing* Λ *as* $\{7.0, -2.0\}$ *and* Υ *as* $\{11.0, 0.0\}$ *through only two transformations of Equation (7). Note that the 2-dimensional MBR by DFTnaive is the same as that by DFTadv. In summary, we significantly reduce the number of transformations from 16 to two while obtaining the same 2-dimensional MBR. This reduction ratio will increase exponentially proportional to the dimension. For example, if we handle 16-dimensional MBRs instead of 4-dimensional ones, the number will be reduced from 65,536 (= 216) to two. Likewise, DFTadv drastically reduce the number of transformations, and based on this reduction it improves the performance significantly as we can see in Section 5.* □

Even though both DFTnaive and DFTadv perform the lower-dimensional MBR transformation correctly, there is a big difference in the number of transformations between DFTnaive and DFTadv. DFTnaive requires $O(2^n)$ transformations for an n-dimensional MBR since it uses all vertex points on the MBR. Thus, for k MBRs, DFTnaive requires $O(k \cdot 2^n)$ lower-dimensional transformations. This exponential complexity, however, is never applicable to actual time-series applications. On the other hand, DFTadv uses just two transformations (Λ and Υ) for an n-dimensional MBR, and thus, it requires only $O(k)$ lower-dimensional transformations for k MBRs. This constant complexity makes DFTadv practically applicable to real time-series matching applications. Figure 3 shows a graph that depicts the numbers of transformations required in DFTnaive and DFTadv, where we set k to 10,000 and change n from four to 512 by multiples of two. As shown in the figure, as the dimension increases, the difference in the number of transformations between DFTnaive and DFTadv becomes drastically larger. Note that the Y axis in the figure has the exponential scale. For example, if the dimension is 16, the ratio between two transformations is 65,536 ($= 2^{16}$), but the ratio reaches 2^{512} if the dimension is 512. This difference in the number of transformations will affect the real execution time, and we will confirm this performance difference in Section 5.

Here, we note that DFTnaive satisfies the equation of $X \in [L, U] \Rightarrow X^{DFT} \in [L, U]^{DFTnaive}$, but does not satisfy its inverse, i.e., the equation of $X \notin [L, U] \Rightarrow X^{DFT} \notin [L, U]^{DFTnaive}$. It means that, even if an HD point X is not contained in an HD MBR $[L, U]$, the transformed LD point X^{DFT} can be contained in the transformed LD MBR $[L, U]^{DFTnaive}$. Also, the inverse means that its contrapositive (i.e., $X^{DFT} \in [L, U]^{DFTnaive} \Rightarrow X \in [L, U]$) is not satisfied. That is, even if X^{DFT} is contained in $[L, U]^{DFTnaive}$,

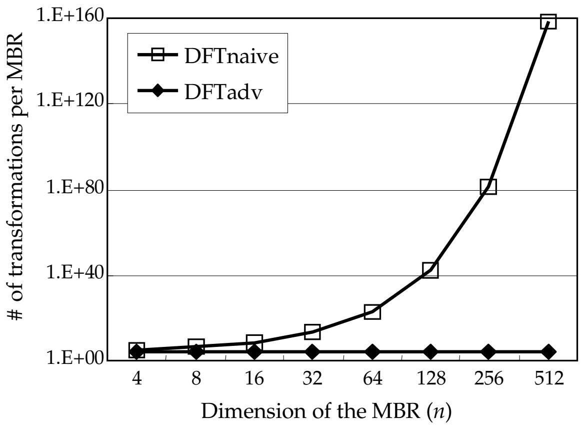


FIGURE 3. Numbers of transformations required in DFTnaive and DFTadv

the original X cannot be contained in the original MBR $[L, U]$. This property is due to that the DFT-based lower-dimensional transformation is not a lossless transformation, i.e., DFTnaive incurs information loss[1] [1, 7]. That is, it is because we cannot restore an exact original HD point from the transformed LD point which already loses some information. DFTadv also has the same property since its MBR is the same as that of DFTnaive. However, both DFTnaive and DFTadv are optimal (i.e., they construct the smallest MBR) among the DFT-based lower-dimensional MBR transformations. Since the proof of optimality can be easily done by Theorems 4.1 and 4.2, we omit the details.

4.2. **DCT-based lower-dimensional MBR transformations.**

In this subsection, we present DCT-based lower-dimensional MBR transformations. Like DFT, we first present a DCT-based naive solution that uses a finite number of vertex points, and then propose an advanced solution that uses a lower-left and an upper-right points only. Definition 4.3 shows a formal definition of the DCT-based naive approach.

Definition 4.3. *For an n-dimensional MBR $[L, U]$, DCTnaive is defined as an operation that constructs an $m(\ll n)$-dimensional MBR by performing DCT-based lower-dimensional transformations on all 2^n vertex points on $[L, U]$.*

Theorem 4.3 guarantees correctness of DCTnaive.

Theorem 4.3. *For an n-dimensional point X and an n-dimensional MBR $[L, U]$, DCTnaive satisfies the equation of $X \in [L, U] \Rightarrow X^{DCT} \in [L, U]^{DCTnaive}$. That is, DCTnaive performs the lower-dimensional MBR transformation correctly.*

Proof: Since $X \in [L, U]$, $l_t \leq x_t \leq u_t$ holds for every t $(0 \leq t \leq n-1)$. Among vertex points on the MBR $[L, U]$, there exist two vertex points P and Q that are constructed by the following Equation (8). In Equation (8), $\theta = \frac{(2t+1)i\pi}{2n}$.

$$p_t = \begin{cases} l_t, & \text{if } \cos\theta \geq 0; \\ u_t, & \text{if } \cos\theta < 0; \end{cases}, \quad q_t = \begin{cases} u_t, & \text{if } \cos\theta \geq 0; \\ l_t, & \text{if } \cos\theta < 0; \end{cases}, \text{ where } 0 \leq t \leq n-1. \tag{8}$$

By the same proof steps in Theorem 4.1, we can show that $\frac{2 \cdot c(i)}{n} \sum_{t=0}^{n-1} p_t \cos\theta$ $(= p_i^{DCT})$ is less than or equal to $\frac{2 \cdot c(i)}{n} \sum_{t=0}^{n-1} x_t \cos\theta$ $(= x_i^{DCT})$, and $\frac{2 \cdot c(i)}{n} \sum_{t=0}^{n-1} q_t \cos\theta$ $(= q_i^{DCT})$ is greater than or equal to $\frac{2 \cdot c(i)}{n} \sum_{t=0}^{n-1} x_t \cos\theta$ $(= x_i^{DCT})$. Therefore, $p_i^{DCT} \leq x_i^{DCT} \leq q_i^{DCT}$ holds for every i. Here, since P^{DCT} and Q^{DCT} are contained in the LD MBR $[L, U]^{DCTnaive}$

[1]DFT itself is lossless, but the DFT-based lower-dimensional transformation that takes a few coefficients after DFT incurs information loss.

by Definition 4.3, X^{DCT} is also contained in $[L, U]^{DCTnaive}$, i.e., $X^{DCT} \in [L, U]^{DCTnaive}$ is satisfied. Therefore, the equation of $X \in [L, U] \Rightarrow X^{DCT} \in [L, U]^{DCTnaive}$ holds. Also, we note that the left-point of $[L, U]^{DCTnaive}$ can be obtained from p_i^{DCT}, and the right-point from q_i^{DCT}. □

Like DFTnaive, however, DCTnaive has a critical problem of incurring a huge number of transformations. Thus, we next propose DCTadv that drastically reduces the number of transformations over DCTnaive. The advanced DFTadv also uses a lower-left and an upper-right points only, and its LD MBR is the same as that of DCTnaive. Definition 4.4 shows a formal definition of DCTadv.

Definition 4.4. *For an n-dimensional MBR $[L, U]$, DCTadv is defined as an operation that constructs an $m(\ll n)$-dimensional MBR of which lower-left and upper-right points are Λ and Υ, respectively, in Equation (9). In Equation (9), $\theta = \frac{(2t+1)i\pi}{2n}$ and $0 \leq i \leq m-1$.*

$$\lambda_i = \frac{2 \cdot c(i)}{n} \sum_{t=0}^{n-1} a_t \cos\theta, \quad \upsilon_i = \frac{2 \cdot c(i)}{n} \sum_{t=0}^{n-1} b_t \cos\theta,$$
$$\text{where } \begin{cases} a_t = l_t, b_t = u_t, & \text{if } \cos\theta \geq 0; \\ a_t = u_t, b_t = l_t, & \text{if } \cos\theta < 0. \end{cases} \tag{9}$$

Theorem 4.4 shows correctness of DCTadv.

Theorem 4.4. *For an n-dimensional point X and an n-dimensional MBR $[L, U]$, DCTadv satisfies the equation of $X \in [L, U] \Rightarrow X^{DCT} \in [L, U]^{DCTadv}$. That is, DCTadv performs the lower-dimensional MBR transformation correctly.*

Proof: In the proof steps of Theorem 4.3, we derived p_i^{DCT} and q_i^{DCT}. We then note that Equation (9) of DCTadv simply denotes p_i^{DCT} as λ_i and q_i^{DCT} as υ_i. Thus, the MBR of DCTadv is the same as that of DCTnaive. Therefore, DCTadv also performs the lower-dimensional MBR transformation correctly since we have already shown correctness of DCTnaive in Theorem 4.3. □

Like the difference in the number of transformations between DFTnaive and DFTadv, there is also a big difference between DCTnaive and DCTadv. That is, for k MBRs, DCTnaive requires $O(k \cdot 2^n)$ transformations, which is exponential and never applicable to real applications; in contrast, DCTadv requires only $O(k)$ transformations, which is constant and easily applicable to real applications. We will also confirm the performance difference in Section 5. Like DFTnaive and DFTadv, the equations of $X \notin [L, U] \Rightarrow X^{DCT} \notin [L, U]^{DCTnaive}$ and $X \notin [L, U] \Rightarrow X^{DCT} \notin [L, U]^{DCTadv}$ are not satisfied. This property is due to that DCTnaive and DCTadv also incur information loss [5].

5. Experimental Evaluation.

In this section we present the results of performance evaluation. We describe the experimental data and environment in Section 5.1 and present the results of the experiments in Section 5.2.

5.1. Experimental data and environment.

The number of transformations and the elapsed time for lower-dimensional MBR transformations are not dependent on types of data sets. Thus, in this paper we use a real stock data set only that consists of 329,112 entries [7, 21]. Each entry of the stock data set contains a bid price and an ask price. Figure 4(a) shows a part of this stock data set, which is used in the traditional similar sequence matching [7, 21, 24][2]. We generate HD MBRs by dividing the whole stock data

[2]The previous works [7, 21, 24] constructed a whole sequence by using only one price value (i.e., either a bid price or an ask price but not both). Thus, we can say that they considered HD points rather than HD MBRs in similar sequence matching.

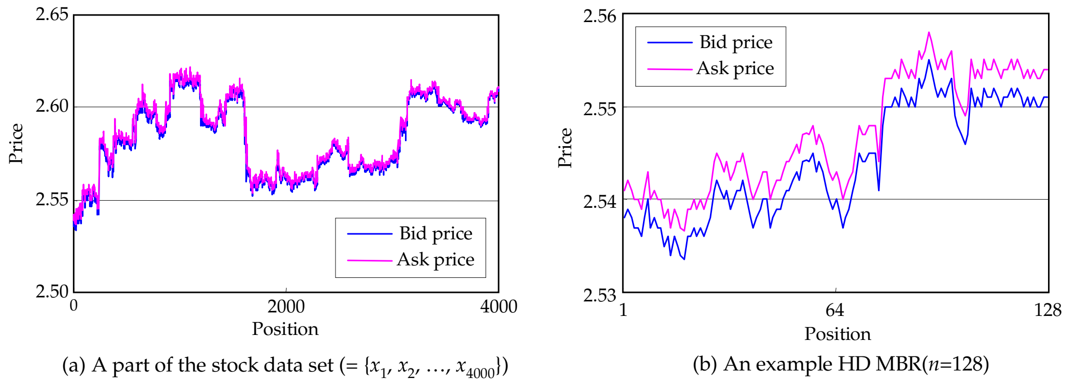


FIGURE 4. A part of experimental data and an example of an HD MBR

set into multiple smaller sequences (i.e., windows in [7, 21]). That is, we divide a long whole sequence into multiple smaller sequences with the dimension wanted. Figure 4(b) shows an example sequence of an HD MBR with the dimension of 128.

In the experiments, we change the dimension n of the HD MBR from four to 512 by multiples of two (i.e., $n = 4, 8, 16, \ldots, 256, 512$). Since the dimension of the LD MBR makes no effect on the experimental results, we set it to a fixed value of two as in [1]. As the experimental methods, we compare DFTnaive and DFTadv as DFT-based solutions and compare DCTnaive and DCTadv as DCT-based solutions. The hardware platform for the experiment is a PC equipped with an Intel Pentium IV 3.20 GHz CPU, 1GB RAM, and a 500.0GB hard disk. The software platform is GNU/Linux Version 2.6.6 operating system. For the experimental results, we measure the number of transformations and the elapsed time for each transformation method.

We also perform additional experiments on subsequence matching [7, 22, 23] to show practical use of the proposed MBR transformation. As we explained in Section 1, subsequence matching is one of important time-series matching applications. For subsequence matching, we use two data sets: one is the stock data of Figure 4, and another is the sine data used in [23]. We discuss these experimental results in Experiments 3 and 4.

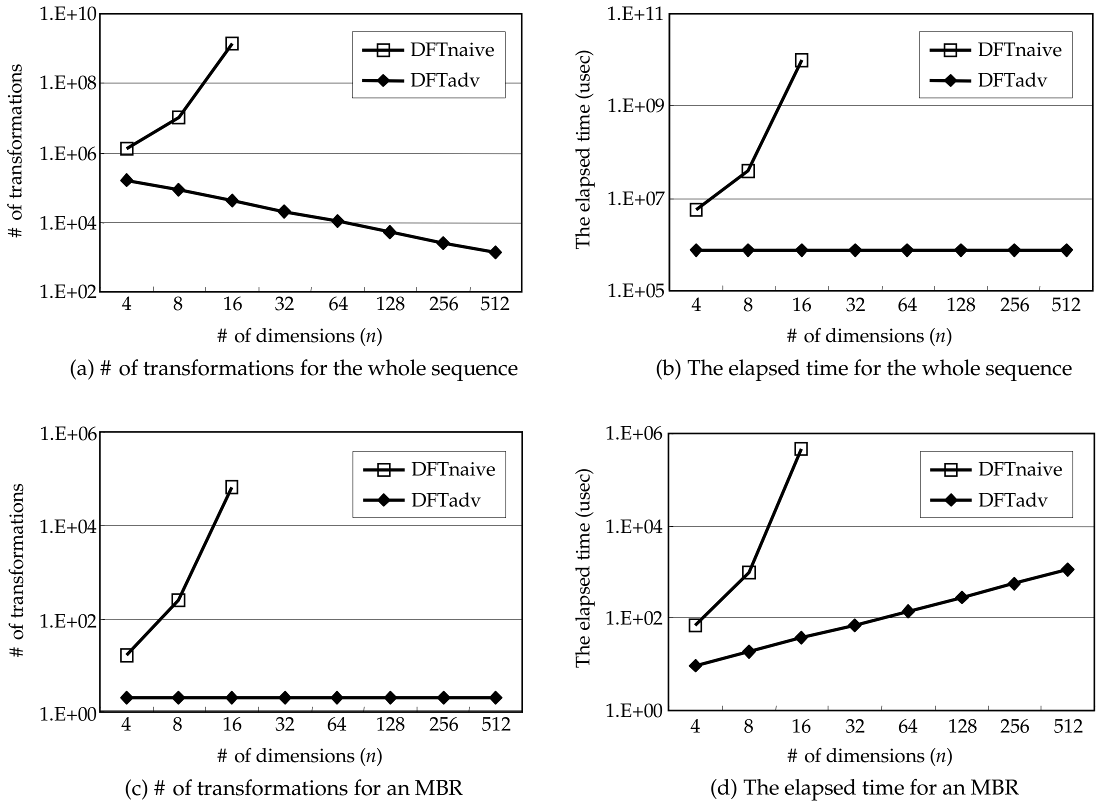
5.2. Experimental results.

In this subsection we explain the experimental results for four MBR transformations. Experiment 1) measures the numbers of transformations and the elapsed times of DFTnaive and DFTadv by varying the dimension n of the HD MBR. Experiment 2) performs the same experiment for DCTnaive and DCTadv.

Experiment 1) DFTnaive vs. DFTadv

Figure 5 shows the experimental results of DFTnaive and DFTadv. In the experiment, as the dimension n of the HD MBR increases, the number k of HD MBRs decreases since k is set to $\lfloor 329,112/n \rfloor$. In the figure, X axis represents the dimension n of the HD MBR, and Y axis the number of transformations or the elapsed time. Figures 5(a) and 5(b) show the results for the whole stock data set, and Figures 5(c) and 5(d) those for one MBR. Note that Y axes in all the four figures have the exponential scale. As shown in the figures, there is no experimental result of DFTnaive for the cases where n is greater than 16. It is because DFTnaive requires a huge number of transformations and takes a too long time if n is greater than 16.

As shown in Figure 5(a), DFTadv significantly reduces the number of transformations over DFTnaive, and the difference becomes larger as n increases. As we have analyzed in Section 4, it is because the number of transformations required in DFTnaive is proportional to $O(k \cdot 2^n)$; in contrast, that in DFTadv to $O(k)$. Next, in Figure 5(a), the number of transformations in DFTadv decreases as n increases. The reason is that, as



(a) # of transformations for the whole sequence

(b) The elapsed time for the whole sequence

(c) # of transformations for an MBR

(d) The elapsed time for an MBR

Figure 5. Experimental results for DFTnaive and DFTadv

n increases, the number k of HD MBRs decreases since $k = \lfloor 329,112/n \rfloor$, but DFTadv requires only two transformations for each MBR. Figure 5(b) shows that DFTadv also reduces the elapsed time significantly over DFTnaive. This performance difference is due to the difference in the number of transformations depicted in Figure 5(a). In Figure 5(b), we note that there is no change in the elapsed time of DFTadv even as n increases. It is because, as the dimension n of the MBR increases, the number of transformations of DFTadv in Figure 5(a) decreases, but the increased dimension requires the more computation time for DFT. In summary of Figures 5(a) and 5(b), when the dimension n is 16, DFTadv reduces the number of transformations by up to 32,800 times and the elapsed time by up to 13,100 times that of DFTnaive. And, this difference between DFTnaive and DFTadv becomes rapidly larger as the dimension increases.

As shown in the results for one MBR in Figures 5(c) and 5(d), DFTadv also significantly outperforms DFTnaive. In Figure 5(c), the number of transformations in DFTnaive rapidly increases in proportion to $O(k \cdot 2^n)$ as n increases. In contrast, DFTadv requires only two transformations for each MBR regardless of n. Thus, Figure 5(c) shows that, as n increases, the difference in the number of transformations between DFTnaive and DFTadv becomes larger. By this difference in the number of transformations, DFTadv reduces the elapsed time over DFTnaive as in Figure 5(d). We note that, in Figure 5(d), the elapsed time of DFTadv becomes larger as n increases. It is because the DFT transformation time for one MBR becomes larger as the dimension n of the MBR increases.

Experiment 2) DCTnaive vs. DCTadv

Figure 6 shows the experimental results of DCTnaive and DCTadv. Since the numbers of transformations required in DCTnaive and DCTadv are identical to those in DFTnaive and DFTadv, respectively, we omit the numbers, but present the elapsed times only as the results. Figure 6(a) shows the result for the whole stock data set, and Figure 6(b)

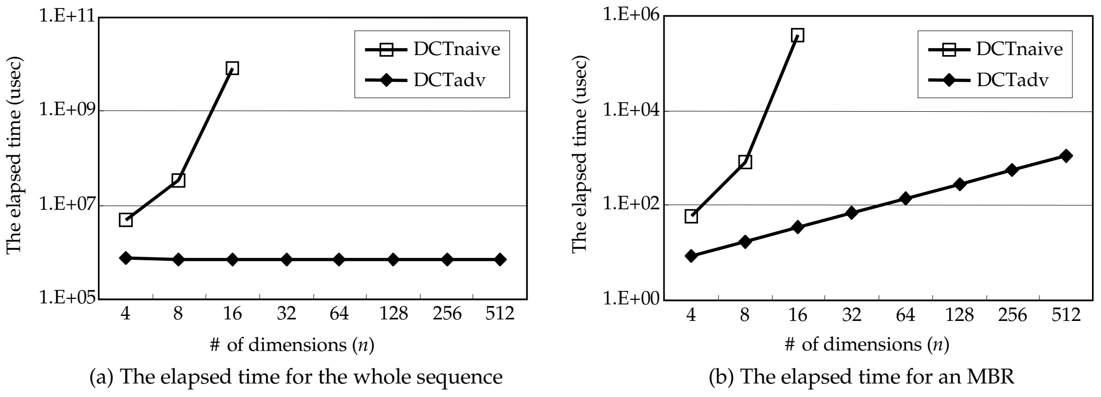


FIGURE 6. Experimental results for DCTnaive and DCTadv

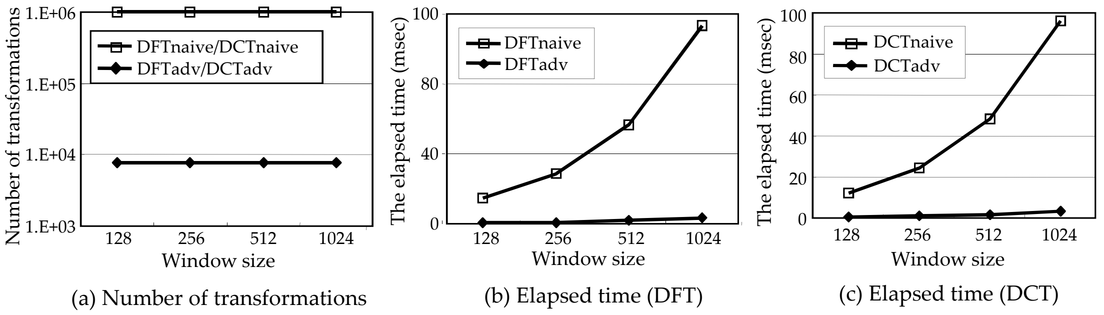


FIGURE 7. Efficiency comparison of lower-dimensional transformations in subsequence matching

that for one MBR. Like DFTnaive, there is no experimental result of DCTnaive for the cases where n is greater than 16. It is also because DCTnaive takes a too long time if n is greater than 16.

The results for DCT solutions in Figures 6(a) and 6(b) are very similar to those for DFT solutions in Figures 5(b) and 5(d). The reason is that the number of transformations is not dependent on types of transformations. That is, it is because the numbers of transformations required in DCTnaive and DCTadv are the same as those in DFTnaive and DFTadv. However, there is a little difference (3% to 6%) in the elapsed time between DFTnaive and DCTnaive (or between DFTadv and DCTadv) since the transformation time by DFT is not exactly same as that by DCT. In summary of Figure 6, when the dimension n is 16, DCTadv reduces the elapsed time by up to 11,600 times that of DCT-naive. And, this difference between DCTnaive and DCTadv becomes rapidly larger as the dimension increases.

Experiment 3) Efficiency of transformations in subsequence matching
In this experiment, we measure the number of transformations and the elapsed time for those transformations in subsequence matching. Figure 7(a) shows the number of lower-dimensional transformations, and Figures 7(b) and 7(c) show the elapsed time per MBR for the naive and advanced transformations. We have fixed the number of subsequences in an MBR to 256, but we have varied the window size [7, 22] from 128 to 1,024. From Figure 7(a), we note that the numbers of transformations do not change even as the window size increases. This is because the numbers are independent of the window size in both transformations. Additionally, in Figures 7(b) and 7(c) we see DFTadv and DCTadv significantly reduce the elapsed time over DFTnaive and DCTnaive, respectively. This

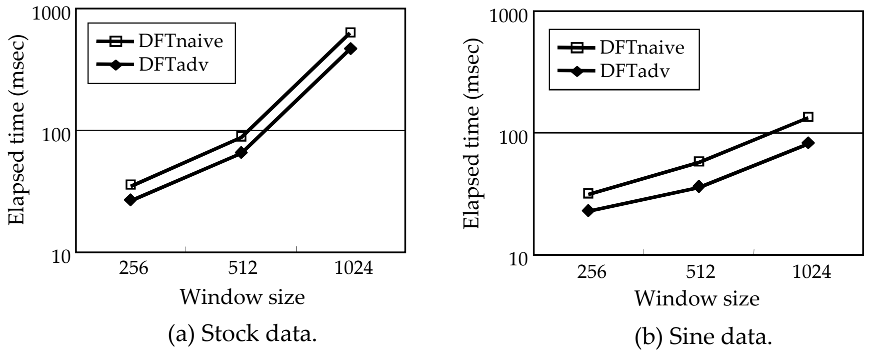


Figure 8. Subsequence matching performance of DFTadv and DFTnaive

result means that our MBR transformations are very suitable as the lower-dimensional transformation of subsequence matching.

Experiment 4) Efficiency on the whole process of subsequence matching
In this experiment, we confirm that the MBR transformation truly improve the whole process of subsequence matching. For this, we perform the actual subsequence matching, which includes lower-dimensional transformations, and measure its actual elapsed time. Figure 8 shows the subsequence matching performance of DFTadv and DFTnaive for varying the window size from 256 to 1024. As shown in the figures, DFTadv improves the matching performance compared with DFTnaive. However, the performance improvement of Figure 8 is relatively small compared with other previous experimental results. This is because the subsequence matching algorithm includes the post-processing step as well as the index-filtering step [1, 7, 22]. That is, much time is required in the post-processing step, and thus the overall performance improvement becomes relatively small. We have performed the same experiment for DCTadv and DCTnaive, but we omit their experimental results here because the results are very similar to those of Figure 8. In conclusion, we can say that our MBR transformations are also applicable to actual subsequence matching applications.

6. Conclusions.

To our knowledge, there has been no research result on the lower-dimensional transformation of *HD MBRs* while there have been many research efforts on the lower-dimensional transformation of *HD points*. However, a lot of real data such as time-series sequences and streaming data can be modeled as an MBR rather than a point in an HD space. To store and search those HD MBRs in a multidimensional index, we need to use the lower-dimensional MBR transformation that converts an HD MBR itself to an LD MBR directly. In this paper, we proposed a formal approach to the lower-dimensional MBR transformation.

We can summarize our work as the following four contributions.

- First, we formally defined the notion of *lower-dimensional MBR transformation*. The lower-dimensional MBR transformation constructs an LD MBR by bounding every LD point to which every possible HD point in the HD MBR is transformed.
- Second, as the naive solutions for DFT and DCT, we proposed DFTnaive and DCTnaive, respectively. DFTnaive and DCTnaive use all the vertex points on an HD MBR rather than an infinite number of points in the MBR. We formally proved their correctness in Theorems 4.1 and 4.3, respectively.
- Third, to solve the problem that the naive solutions require a huge number of transformations, we proposed DFTadv and DCTadv as the advanced solutions. DFTadv and DCTadv use only two points (a lower-left and an upper-right points) rather

than all vertex points on an MBR. We also formally proved their correctness in Theorems 4.2 and 4.4, respectively.

- Fourth, through analysis and experiments, we showed superiority of DFTadv and DCTadv. By formally deriving the number of transformations, we analytically show that the complexity for performing MBR transformations is drastically reduced from exponential ($O(2^n)$) to constant ($O(1)$). We then empirically showed that, compared with the naive solutions, DFTadv and DCTadv significantly reduced the execution time by up to 13,100 and 11,600 times, respectively, when the dimension was 16. More importantly, the larger dimension would cause the more performance improvement.

These results indicate that the proposed notion of lower-dimensional MBR transformation, DFT-based solutions, and DCT-based solutions provide a useful framework for a variety of applications that use HD MBRs.

Acknowledgement. This work was partially supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract.

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, IL, USA, pp.69-84, 1993.

[2] M. Barrena, E. Jurado, P. Marquez-Neila and C. Pachon, A flexible framework to ease nearest neighbor search in multidimensional data spaces, *Data and Knowledge Engineering*, vol.69, no.1, pp.116-136, 2010.

[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] 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. on Knowledge and Data Engineering*, vol.15, no.3, pp.686-705, 2003.

[6] M.-J. Choi, H.-S. Kim and Y.-S. Moon, Publishing sensitive time-series data under preservation of privacy and distance orders, *International Journal of Innovative Computing, Information and Control*, vol.8, no.5(B), pp.3619-3638, 2012.

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

[8] A. Gutmann, R-trees: A dynamic index structure for spatial searching, *Proc. of Int'l Conf. on Management of Data*, Boston, MA, USA, pp.47-57, 1984.

[9] R. H. Guting, An introduction to spatial database systems, *The VLDB Journal*, vol.3, no.4, pp.357-399, 1994.

[10] C. Hamzacebi, Improving artificial neural networks' performance in seasonal time series forecasting, *Information Sciences*, vol.178, no.23, pp.4550-4559, 2008.

[11] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 2nd Edition, Morgan Kaufmann Publishers, 2006.

[12] W.-S. Han, J. Lee, Y.-S. Moon, S. Hwang and H. Yu, A new approach for processing ranked subsequence matching based on ranked union, *Proc. of Int'l Conf. on Management of Data*, Athens, Greece, pp.457-468, 2011.

[13] G. R. Hjaltason and H. Samet, Incremental similarity search in multimedia databases, *Technical Report 4199*, Dept. of Computer Science, University of Maryland, College Park, 2000.

[14] M. J. Hsieh, M. S. Chen and P. S. Yu, Integrating DCT and DWT for approximating cube streams, *Proc. of the 14th ACM Int'l Conf. on Information and Knowledge Management*, Bremen, Germany, pp.179-186, 2005.

[15] E. J. Keogh, K. Chakrabarti, S. Mehrotra and M. J. Pazzani, Locally adaptive dimensionality reduction for indexing large time series databases, *Proc. of Int'l Conf. on Management of Data*, Santa Barbara, CA, USA, pp.151-162, 2001.

[16] B.-S. Kim, Y.-S. Moon and J. Kim, Noise control boundary image matching using time-series moving average transform, *Proc. of the 18th Int'l Conf. on Database and Expert Systems Applications*, Turin, Italy, pp.362-375, 2008.

[17] S.-P. Kim, Y.-S. Moon and S.-K. Hong, An envelope-based approach to rotation-invariant boundary image matching, *Proc. of the 13th Int'l Conf. on Data Warehousing and Knowledge Discovery*, Toulouse, France, pp.382-393, 2011.

[18] F. Korn, H. V. Jagadish and C. Faloutsos, Efficiently supporting Ad Hoc queries in large datasets of time sequences, *Proc. of Int'l Conf. on Management of Data*, Tucson, AZ, USA, pp.289-300, 1997.

[19] X. Lian and L. Chen, Efficient similarity search over future stream time series, *IEEE Trans. on Knowledge and Data Engineering*, vol.20, no.1, pp.40-54, 2008.

[20] W.-K. Loh, Y.-S. Moon and J. Srivastava, Distortion-free predictive streaming time-series matching, *Information Sciences*, vol.180, no.8, pp.1458-1476, 2010.

[21] Y.-S. Moon, K.-Y. Whang and W.-K. Loh, Duality-based subsequence matching in time-series databases, *Proc. the 17th Int'l Conf. on Data Engineering*, Germany, pp.263-272, 2001.

[22] 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, WI, USA, pp.382-393, 2002.

[23] Y.-S. Moon, An MBR-safe transform for high-dimensional MBRs in similar sequence matching, *Proc. of the 12th Int'l Conf. on Database Systems for Advanced Applications*, Bangkok, Thailand, pp.79-90, 2007.

[24] 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.

[25] 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.

[26] Y.-S. Moon, H.-S. Kim and S.-P. Kim, Noise averaging effect of privacy-preserving data mining in time-series databases, *ICIC Express Letters*, vol.5, no.2, pp.285-291, 2011.

[27] A. Natsev, R. Rastogi and K. Shim, WALRUS: A similarity retrieval algorithm for image databases, *IEEE Trans. on Knowledge and Data Engineering*, vol.16, no.3, pp.301-316, 2004.

[28] W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd Edition, Cambridge University Press, 1992.

[29] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*, 3rd Edition, Academic Press, 2006.

[30] H. Wu, B. Salzberg and D. Zhang, Online event-driven subsequence matching over financial data streams, *Proc. of Management of Data*, Paris, France, pp.23-34, 2004.

[31] D. Zhao, W. Gao and Y. K. Chan, Morphological representation of DCT coefficients for image compression, *IEEE Trans. on Circuits and Systems for Video Technology*, vol.12, no.9, pp.819-823, 2002.