PAPER

Flexible and Fast Similarity Search for Enriched Trajectories*

Hideaki OHASHI[†a)], **Toshiyuki SHIMIZU**[†], ***Nonmembers***, ***and*** **Masatoshi YOSHIKAWA**[†], ***Member***

SUMMARY In this study, we focus on a method to search for similar trajectories. In the majority of previous works on searching for similar trajectories, only *raw* trajectory data were used. However, to obtain deeper insights, additional time-dependent trajectory features should be utilized depending on the search intent. For instance, to identify similar combination plays in soccer games, such additional features include the movements of the team players. In this paper, we develop a framework to flexibly search for similar trajectories associated with time-dependent features, which we call *enriched trajectories*. In this framework, *weights*, which represent the relative importance of each feature, can be flexibly given by users. Moreover, to facilitate fast searching, we first propose a lower bounding measure of the DTW distance between enriched trajectories, and then we propose algorithms based on this lower bounding measure. We evaluate the effectiveness of the lower bounding measure and compare the performances of the algorithms under various conditions using soccer data and synthetic data. Our experimental results suggest that the proposed lower bounding measure is superior to the existing measure, and one of the proposed algorithms, which is based on the threshold algorithm, is suitable for practical use.

key words: *trajectories, similarity search, dynamic time warping*

1. Introduction

In recent years, advances in location-acquisition techniques have resulted in the generation of many types of trajectory data, such as hurricane tracking data [1], vessel motion data [2] and sports tracking data [3], [4]. Each of these datasets can be analyzed for recommendations, predictions, and event detection. Numerous studies have introduced various analysis methods for many types of trajectories, such as clustering [1] and outlier detection [5]. In this study, we focus on a search for similar trajectories, which can be applied to many of the above types of analyses.

The majority of methods for similar trajectory search use only *raw* trajectory data [6], [7]; however, to obtain deeper insights, additional time-dependent features should be utilized depending on the search intent. For example, if you wish to identify similar hurricanes, such additional features include the correlations between their speeds and their atmospheric pressures. In addition, if one wishes to find vessels that are moving under similar circumstances, such additional features include the weather and ocean current data. In this paper, we propose a framework to flexibly search for similar trajectories associated with time-dependent features, which we call *enriched trajectories*. In this framework, a user can flexibly set the *weights*, which represent the relative importance of each feature. Although there are several studies that address time-dependent features [8]–[11], our work has novelty in the flexible search by inputting weights according to the search intent.

In this study, we utilize dynamic time warping (DTW) [12] as a measure of the distance between enriched trajectories. DTW is a distance measure for time-series data that is simple and accurate [13]. However, the computational cost of DTW is high; therefore, many techniques have been proposed to accelerate DTW-based similarity searches. In particular, many techniques with lower bounding measures have been proposed [7], [14], [15].

In this paper, to achieve fast searching, we propose a lower bounding measure that is specifically designed for our proposed framework. The computational cost of this lower bounding measure is very low. In addition, we propose four algorithms based on the lower bounding measure for fast similarity searching. We name the four algorithms naive lower bounding (NLB), sorted lower bounding (SLB), partially selected lower bounding (PSLB), and threshold algorithm lower bounding (TALB). The NLB algorithm uses the lower bounding measure to prune out candidates and serves as the basis for the other three algorithms. SLB is an algorithm in which candidates are sorted by the lower bound distance. PSLB is an algorithm in which candidates are partially selected by the lower bound distance. Finally, TALB is an algorithm that is based on the threshold algorithm [16].

We compare the performances of the proposed lower bounding measure and the existing measure for multivariate time series [17] and confirm that our proposed measure is superior to the existing measure. In addition, we compare the performances of the four algorithms under various conditions using soccer data and synthetic data. Our results reveal that PSLB is slightly superior in most cases. However, the TALB algorithm is considerably superior when both the number of trajectories is large and the average length of the trajectories is short or when the search allocates a larger weight to one feature of the trajectories.

The contributions of this paper can be summarized as follows:

Manuscript received December 6, 2016.
Manuscript revised March 22, 2017.
Manuscript publicized May 30, 2017.

[†]The authors are with the Dept. of Social Informatics, Graduate School of Informatics, Kyoto University, Kyoto-shi, 606–8501 Japan.

*A preliminary version of this paper was accepted in the 11th International Workshop on Spatial and Spatiotemporal Data Mining (SSTDM-16) in cooperation with IEEE/ICDM 2016, December 2016.

a) E-mail: ohashi@db.soc.i.kyoto-u.ac.jp

DOI: 10.1587/transinf.2016EDP7482

- We propose a framework for flexible similarity searches for enriched trajectories considering the search intent.
- We propose a lower bounding measure that utilizes the characteristics of the proposed framework.
- We propose four algorithms using the proposed lower bounding measure and compare their performances.

The remainder of this paper is structured as follows: Sect. 2 reviews the related work. The problem statement is discussed in Sect. 3. In Sect. 4, we propose a novel lower bounding measure and four algorithms for fast similarity searches of enriched trajectories. We present our experimental results in Sect. 5. In Sect. 6, we conclude our work with a discussion of future work.

2. Related Work

In this section, we introduce similarity measures for time-series data, including trajectory data. Additionally, we introduce the previous studies that address the time-dependent features of an object's trajectory. We consider that the time-dependent features can be classified into two types. The first type includes the features that are derived from the primitive features of the trajectory, such as the object's speed and direction. The second type encompasses the features that are obtained by combining primitive features with other data sources, such as the positional relationships between the object and another object. We call the former *intrinsic features* and the latter *extrinsic features*. There is a long history of studies addressing intrinsic features [8], [9]; however, the number of studies that consider extrinsic features has recently been increasing because of the increasing amount of attention being focused on cross-domain data fusion [18]. In this paper, both types of features are treated identically. In contrast to previous studies, our research addresses both types of features and their relative importance.

2.1 Distance Measures

Dynamic time warping (DTW) [12] is the most widely used distance measure for time series. DTW is an algorithm that allows some points to be repeated to minimize the sum of the distance between points, which suits the characteristics of trajectories as follows:

- Two trajectories need not be observed synchronously for the similarity between them to be measured.
- Similar trajectory patterns often appear in different regions.

DTW is simple and accurate [13]; however, the computational cost of DTW is high. Accordingly, many attempts have been made to reduce the computational cost of DTW [7], [14], [15], [19], [20]. Rath et al. proposed the lower bound *LB_MV* for the DTW distance of multivariate time series [17]. This is an extension of the lower bound *LB_Keogh* [14] for the DTW distance of univariate time series. Some methods other than those using lower bounds have been proposed for accelerating the similarity search of time series data using DTW, such as early abandon [19] and data abstraction [20], which can be extended to the top-k problem of the enriched trajectories. However, all of these methods can be combined with techniques using lower bounds. In addition, to the best of our knowledge, only the research of [17] proposed a lower bound that can be extended to the enriched trajectories. Thus, it is considered reasonable that we compare the performances of the proposed lower bound and *LB_MV*. We confirm that our proposed lower bound is superior to *LB_MV* in Sect. 5.2.1.

Lee et al. [1] defined the distance function between trajectory segments as a linear combination of three components with weights. In this study, we similarly define the distance function as a linear combination with weights, and we utilize the weights as the relative importance of each feature.

2.2 Intrinsic Features

Pelekis et al. [8] introduced a similarity search framework for application to a trajectory database that consists of a set of distance operators based on both primitive (space and time) and derived (speed and direction) parameters of trajectories. Buchin et al. [9] defined many criteria under which a trajectory can be homogeneous, including location, heading, speed, velocity, curvature, sinuosity, and curviness, and they presented a framework for segmenting a trajectory based on these criteria.

Note that some people might think that intrinsic features, such as speeds and directions, are redundant because DTW can measure the distance between sequences in which they vary. However, intrinsic features are important for cases such as those in which we want to retrieve enriched trajectories that are spatially far away from each other but have similar speed patterns.

2.3 Extrinsic Features

Zheng et al. [10] studied the problem of efficient similarity searches for trajectories associated with activity information that is generated from location-based web applications. Zheng et al. [11] generated air quality inferences based on the trajectories of vehicles, POIs, and meteorological data.

3. Problem Statement

In this section, we present the problem statement and the necessary definitions. Moreover, we show an example of similar enriched trajectories that suggests our proposed framework is useful. In our proposed framework, a user selects a query from a dataset of enriched trajectories and inputs weights; then, the top-k results that are most similar to the query are returned. The weights represent the relative importance of features in an enriched trajectory. The definitions of enriched trajectories and the distance between them

are provided below.

3.1 Enriched Trajectory

In this study, a trajectory is defined as follows:

$$p^i = \langle p^i(1), p^i(2), \ldots, p^i(n) \rangle$$

where i is the identification index, n is the length of p^i, and $p^i(j)$ $(j = 1, 2, \ldots, n)$ denote spatial points. We assume that all temporal intervals between adjacent points are equal in this study.

We refer to a tuple of spatial points and time-dependent feature values as an *enriched point*. The j-th enriched point of p^i is defined as follows:

$$ep^i(j) = (p^i(j), fv_1^i(j), \ldots, fv_m^i(j))$$

where $fv_l^i(j)$ $(l = 1, 2, \ldots, m)$ denote time-dependent feature values and m is the number of time-dependent feature values. Thus, we represent an enriched trajectory as follows:

$$ep^i = \langle ep^i(1), ep^i(2), \ldots, ep^i(n) \rangle$$

3.2 Distance between Enriched Trajectories

In this study, we utilize DTW to calculate the distance between enriched trajectories. First, we introduce the distance between two enriched points:

$$\begin{aligned} &EDist(ep^i(j), ep^g(h)) \\ &\quad = w_p \cdot D_p(p^i(j), p^g(h)) \\ &\qquad + \sum_{l=1}^{m} w_{fv_l} \cdot D_{fv_l}(fv_l^i(j), fv_l^g(h)) \end{aligned} \tag{1}$$

where $w_p, w_{fv_1}, \ldots, w_{fv_m}$ are the weights given by the user, in which all weights are non-negative real numbers; D_p denotes the distance function for spatial points; and $D_{fv_1}, \ldots, D_{fv_m}$ denote the distance function for time-dependent feature values.

Note that it is difficult to manually input all the weights accurately and obtain the desired insights without any support. Some approaches with learning techniques [21] or statistical techniques [22] could support users in selecting proper weights; however, solving this problem is beyond the scope of this paper.

Next, we present the DTW algorithm. Let $p^i = \langle p^i(1), p^i(2), \ldots, p^i(s) \rangle$, and $p^g = \langle p^g(1), p^g(2) \ldots, p^g(t) \rangle$ be two trajectories. DTW is defined as follows:

$$\begin{aligned} &DTW(p^i, p^g) = f(s, t) \\ &f(j, h) = D(j, h) + min \begin{cases} f(j-1, h) \\ f(j, h-1) \\ f(j-1, h-1) \end{cases} \\ &f(0, 0) = 0, f(j, 0) = f(0, h) = \infty \\ &(j = 1, 2, \ldots, s; h = 1, 2, \ldots, t) \end{aligned}$$

where $D(j, h)$ denotes the distance between $p^i(j)$ and $p^g(h)$.

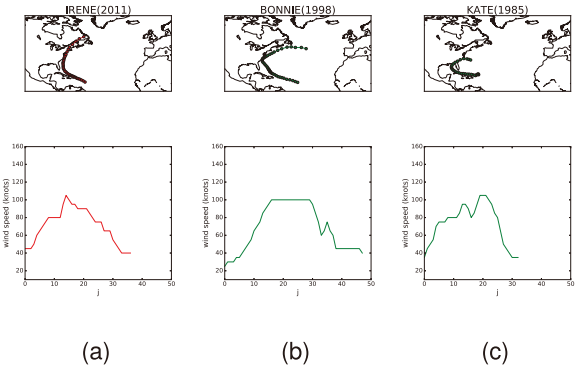


Fig. 1 An example of similar enriched trajectories, which is composed of a hurricane trajectory and its wind speed. (a) Query. (b) Most similar enriched trajectory when $w_p = 3$ and $w_{fv_1} = 1$. (c) Most similar enriched trajectory when $w_p = 1$ and $w_{fv_1} = 3$.

DTW is an algorithm that allows some points to be repeated to achieve the best alignment. When we compute $DTW(ep^i, ep^g)$, $D(j, h)$ is equal to $EDist(ep^i(j), ep^g(h))$ (Eq. (1)).

3.3 An Example of Similar Enriched Trajectories

We present an example of similar enriched trajectories in Fig. 1. We utilized 451 Atlantic hurricane tracking data, which are recorded every 6 hours, in Unisys Weather[†]. In this example, we consider the hurricane trajectories to be associated with the wind speed. When we input hurricane IRENE (Fig. 1 (a)) that occurred in 2011 and set the weight of the trajectories (w_p) to 3 and the weight of the wind speed (w_{fv_1}) to 1, BONNIE (Fig. 1 (b)) was obtained as the most similar enriched trajectory. Additionally, KATE (Fig. 1 (c)) is obtained as the most similar enriched trajectory when we set the weight of trajectories (w_p) to 1 and the weight of wind speed (w_{fv_1}) to 3. This figure shows that the trajectory of IRENE is similar to that of BONNIE and that the sequence of wind speed of IRENE is similar to that of KATE. Therefore, this example suggests that we can obtain the desired result by inputting weights according to the search intent.

4. Algorithms

In our proposed framework, a user can obtain their desired results by modifying the specified weights. However, because of the high computational cost of DTW, searching the desired results requires too much time. Thus, we propose a lower bounding measure that is specifically designed for our proposed framework. The cost of computing this lower bounding measure is very low. Moreover, we propose four algorithms based on this lower bounding measure.

4.1 Proposed Lower Bounding Measure

We propose a lower bounding measure based on the following theorem.

[†]http://weather.unisys.com/hurricane/

Theorem 1: Let $w_p, w_{fv_1}, w_{fv_2}, \ldots, w_{fv_m}$ be the weights used to compute the distance between enriched trajectories. Then,

$$DTW(ep^i, ep^g) \geq w_p \cdot DTW(p^i, p^g) + \sum_{l=1}^{m} w_{fv_l} \cdot DTW(fv_l^i, fv_l^g)$$

Theorem 1 states that a lower bound on the DTW distance between two enriched trajectories can be obtained as a linear combination of the DTW distance between the two corresponding trajectories and the DTW distances between the corresponding time-dependent feature sequences.

Before proving Theorem 1, we verify its validity through an example. For simplicity, we consider a pair of enriched trajectories, i.e., trajectories associated with time-dependent feature sequences, whose weights $(w_p, w_{fv_1}) = (0.5, 0.5)$. Figure 2 depicts (a) the distance matrix between p^1 and p^2, (b) that between fv_1^1 and fv_1^2 and (c) that between ep^1 and ep^2. In addition, each red path in Fig. 2 indicates an optimal warping path that represents a DTW alignment between the two sequences. In this example, the DTW distance between the trajectories is 15, that between the time-dependent feature sequences is 22 and that between the enriched trajectories is 20. Thus, it can be observed that $20 \geq 0.5 \cdot 15 + 0.5 \cdot 22$ satisfies Theorem 1.

Proof 1: For $E1, E2 \in \{ep, p, fv_1, \ldots, fv_m\}$, let WP_{E1} be the warping path in $E1$'s distance matrix, and let $WDist_{E2}(WP_{E1})$ be the sum of the costs of $E2$'s grid cells on WP_{E1}. DTW chooses a path that minimizes the sum of the costs of $E1$'s grid cells; therefore, the following inequalities are satisfied:

$$\begin{aligned} &WDist_p(WP_{ep}) \geq WDist_p(WP_p) \\ &WDist_{fv_1}(WP_{ep}) \geq WDist_{fv_1}(WP_{fv_1}) \\ &\qquad\vdots \\ &WDist_{fv_m}(WP_{ep}) \geq WDist_{fv_m}(WP_{fv_m}) \end{aligned}$$

$p^1(4)$	4	1	7	6
$p^1(3)$	1	3	5	2
$p^1(2)$	3	5	4	3
$p^1(1)$	1	2	11	3
	$p^2(1)$	$p^2(2)$	$p^2(3)$	$p^2(4)$

(a) Trajectories.

$fv_1^1(4)$	12	6	9	6
$fv_1^1(3)$	8	5	14	4
$fv_1^1(2)$	1	2	12	3
$fv_1^1(1)$	1	3	6	6
	$fv_1^2(1)$	$fv_1^2(2)$	$fv_1^2(3)$	$fv_1^2(4)$

(b) Time-dependent feature sequences.

$ep^1(4)$	8	3.5	8	6
$ep^1(3)$	4.5	4	9.5	3
$ep^1(2)$	2	3.5	8	3
$ep^1(1)$	1	2.5	8.5	4.5
	$ep^2(1)$	$ep^2(2)$	$ep^2(3)$	$ep^2(4)$

(c) Enriched trajectories.

Fig. 2 Warping paths.

Thus, from the above inequalities and Eq. (1),

$$\begin{aligned} &DTW(ep^i, ep^g) \\ &= WDist_{ep}(WP_{ep}) \\ &= w_p \cdot WDist_p(WP_{ep}) + \sum_{l=1}^{m} w_{fv_l} \cdot WDist_{fv_l}(WP_{ep}) \\ &\geq w_p \cdot WDist_p(WP_p) + \sum_{l=1}^{m} w_{fv_l} \cdot WDist_{fv_l}(WP_{fv_l}) \\ &= w_p \cdot DTW(p^i, p^g) + \sum_{l=1}^{m} w_{fv_l} \cdot DTW(fv_l^i, fv_l^g) \end{aligned}$$

Thus, we complete the proof. □

4.2 Fast Top-k Search Algorithms Based on the Proposed Lower Bound

We propose the following four algorithms that utilize the right-hand side of the inequality in Theorem 1 as the lower bound on the DTW distance between two enriched trajectories:

1. Naive lower bounding (NLB)
2. Sorted lower bounding (SLB)
3. Partially selected lower bounding (PSLB)
4. Threshold algorithm lower bounding (TALB)

4.2.1 Naive Lower Bounding

This simple algorithm serves as the basis for the following three algorithms. The idea of this algorithm is to use the lower bounding measure to prune out candidate enriched trajectories whose lower bounds are greater than the k-th DTW distance. First, we preserve the DTW distances between all pairs of trajectories and between all pairs of time-dependent feature sequences in a data structure ($preDTW$), as shown in Algorithm 1. This procedure, called the PREPROCESS PHASE, is completed before the user inputs the desired weight and selects a query. Second, we search for the k most similar enriched trajectories using the lower bounds (SEARCH PHASE), as shown in Algorithm 2. We store the identification indices i ($i = 1, \ldots, k$) of the k candidates and the DTW distances between the query ep^q and ep^i

Algorithm 1 Making Data Structure for the Search (PREPROCESS PHASE)

INPUT: $\{ep^1, \ldots, ep^N\}$ ▷ $ep^i = (p^i, fv_1^i, \ldots, fv_m^i)$
OUTPUT: data structure $preDTW$

```
for i ← 1 to N do
    for j ← 1 to N do
        preDTW[i][j].p ← DTW(p^i, p^j)
        for l ← 1 to m do
            preDTW[i][j].fv_l ← DTW(fv_l^i, fv_l^j)
        end for
    end for
end for
    return preDTW
```

Algorithm 2 Naive Search (SEARCH PHASE)

INPUT: $\{ep^1, \ldots, ep^N\}$ ▷ $ep^i = (p^i, fv_1^i, \ldots, fv_m^i)$
INPUT: query index $q \in \{1 \ldots N\}$
INPUT: $W = \{w_p, w_{fv_1}, \ldots, w_{fv_m}\}$
INPUT: k ▷ the number of outputs
INPUT: data structure $preDTW$
OUTPUT: the indices of the top-k outputs most similar to ep^q

```
for i ← 1 to k do                                   ▷ except i ← q
    PQ.push([i, DTW(ep^q, ep^i)])
end for
for i ← k + 1 to N do                               ▷ except i ← q
    lb ← lower_bound(preDTW[q][i], W)
    if lb < PQ.top.dtw then
        true_dist ← DTW(ep^q, ep^i)
        if true_dist < PQ.top.dtw then
            PQ.pop()
            PQ.push([i, DTW(ep^q, ep^i)])
        end if
    end if
end for
return PQ.index
```

in a priority queue (PQ) ranked by the DTW distance (the pop function returns the largest element). Then, we scan the remaining candidates ep^i $(i = k+1, \ldots, n)$. During this scan, we calculate the lower bounds between ep^q and ep^i using $preDTW$ and the weights W. If a lower bound is less than the k-th DTW distance in the priority queue, then we calculate the DTW distance between ep^q and ep^i; otherwise, we prune out this calculation. Furthermore, if the DTW distance between ep^q and ep^i is greater than the k-th DTW distance in the priority queue, then we pop the top of the priority queue and push in a set consisting of i and the DTW distance between ep^q and ep^i. Let N be the number of candidates, and let δ_1 be the number of times that the DTW algorithm is executed after the initial k candidates are stored. Then, the number of lower bounds calculated is $N - k$, and the number of DTW executions is $k + \delta_1$. We discuss the detailed computational costs of the proposed algorithm in Sect. 4.2.5.

4.2.2 Sorted Lower Bounding

We expect that the DTW distance between a candidate and the query should be small if its lower bound is small. In addition, we expect that the earlier we can find a small DTW distance, the more candidates that we will prune out. This algorithm computes the lower bounds between the query and all candidates and then sorts all of these lower bounds using quicksort immediately after the weights are input. Then, the similarity search begins from the candidates with the smallest lower bounds. Let δ_2 be the number of times that the DTW algorithm is executed after the initial k candidates are stored. Then, the number of lower bounds calculated is N, and the number of DTW executions is $k + \delta_2$, where $\delta_2 \leq \delta_1$ in most cases.

4.2.3 Partially Selected Lower Bounding

The algorithm introduced above cannot perform efficiently when there are many candidates, namely, when N is large. This is because too much time is required to sort all the lower bounds. This algorithm utilizes quickselect [23], which is a partition-based selection algorithm that achieves linear performance, rather than quicksort. Quickselect can select the top-k candidates whose lower bounds are small more rapidly than can quicksort. However, quickselect cannot sort the remaining candidates; therefore, the pruning capability of this algorithm is lower than that of the previous algorithm. Let δ_3 be the number of times that the DTW algorithm is executed after the initial k candidates are stored. Then, the number of lower bounds calculated is N, and the number of DTW executions is $k + \delta_3$, where $\delta_2 \leq \delta_3 \leq \delta_1$ in most cases.

4.2.4 Threshold Algorithm Lower Bounding

We propose an algorithm that utilizes the characteristics of our proposed framework. Fagin's threshold algorithm [16] is a top-k combination algorithm that achieves fast searching with the lists sorted by each feature. In this algorithm, we prepare the sorted lists of the DTW distances between trajectories and between time-dependent feature sequences in the PREPROCESS PHASE. This algorithm is described as follows:

1. Execute sorted access in parallel on each of the sorted lists until k enriched trajectories have been observed. When any enriched trajectory ep^i is observed in a list for the first time, compute the DTW distance between ep^q and ep^i and remember ep^i and its DTW distance.
2. Access the next candidates of the sorted lists. Define a threshold value as the linear combination of each DTW distance of the candidates with the input weights. If at least k enriched trajectories have been observed whose DTW distances are less than or equal to the threshold value, then halt. The outputs are the k enriched trajectories that have been observed with the lowest DTW distances.
3. When ep^i is observed during the sorted access process in a list for the first time, randomly access the other lists to find every DTW distance between p^q and p^i and between fv_l^q and fv_l^i. Then, compute the lower bound, that is, compute the linear combination of these distances with the input weights. If the lower bound is less than any DTW distance between the enriched trajectories observed thus far, then compute the DTW distance between ep^q and ep^i and remember ep^i and its DTW distance. Then, return to step 2 to continue sorted access.

Here, we present an example. Let ep^q be a query, let $ep^1 \ldots ep^5$ be the candidates, and let fv_1 and fv_2 be time-dependent features. We assume that the problem of in-

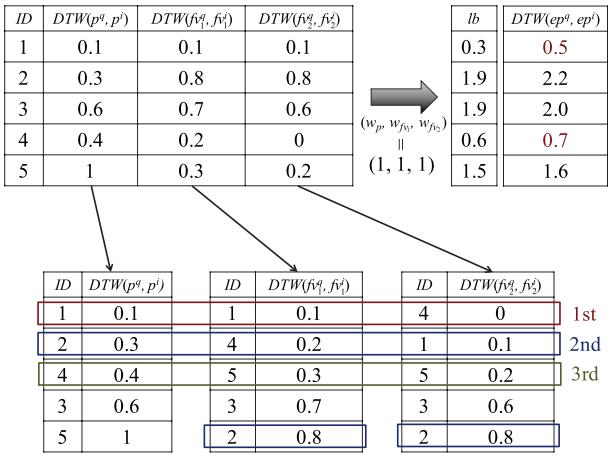


Fig. 3 The TALB algorithm (based on a threshold algorithm).

terest is to search for the top 2 similar enriched trajectories. The upper left of Fig. 3 shows the DTW distances between p^q and p^i, between fv_1^q and fv_1^i, and between fv_2^q and fv_2^i. In addition, the upper right of Fig. 3 shows the lower bounds and the DTW distances between ep^q and ep^i; in this case, the top 2 similar enriched trajectories are ep^1 and ep^4. Moreover, the bottom of Fig. 3 presents lists of the DTW distances between trajectories and between time-dependent feature sequences. During the process of accessing the first items from every list, called the first round, the algorithm computes the DTW distances of ep^1 (0.5) and ep^4 (0.7). In the second round, the algorithm first computes the threshold ($1 \cdot 0.3 + 1 \cdot 0.2 + 1 \cdot 0.1 = 0.6$), which is less than the DTW distance of ep^4 (0.7); therefore, the algorithm continues. Next, the lower bound on ep^2, which has been observed for the first time, is computed, and this lower bound (1.9) is greater than the DTW distance of ep^4; therefore, the DTW distance of ep^2 is not computed. In the third round, the threshold ($1 \cdot 0.4 + 1 \cdot 0.3 + 1 \cdot 0.2 = 0.9$) is greater than any DTW distance computed thus far; therefore, the algorithm is halted.

This algorithm is effective when there are too many enriched trajectories for quickselect to be applicable. For this algorithm, it is necessary to prepare sorted lists of the DTW distances between trajectories and between time-dependent feature sequences; however, this procedure can be completed during the PREPROCESS PHASE. Let $N_{TA} (\leq N)$ be the number of lower bounds calculated, and let δ_4 be the number of times that the DTW algorithm is executed after the initial k candidates are stored. Then, the number of lower bounds calculated is N_{TA}, and the number of DTW executions is $k + \delta_4$, where $\delta_2 \leq \delta_3 \leq \delta_4 \leq \delta_1$ in most cases. In addition, if we consider access scheduling [24], then this algorithm can operate more efficiently. Determining the optimal scheduling for this algorithm will be a task for future research.

4.2.5 Computational Costs of the Proposed Algorithms

Let m be the number of time-dependent feature values, let n be the average length of the enriched trajectories, and let N be the number of enriched trajectories. Then, the average computational times are as shown in Table 1. The average time required for calculating the lower bounds (*lb calculation*) is the number of lower bounds calculated multiplied by m because each lower bound is computed as a linear combination of the DTW distance between the trajectories and the DTW distances between the time-dependent feature sequences. The average time required for calculating the DTW distances (*DTW calculation*) is the average number of calculated DTW distances multiplied by n^2 because computing the DTW distance has a complexity of n^2. The average *lb preparation* performance of the SLB algorithm is the average performance for the sorting of the lower bounds using quicksort. Similarly, the average *lb preparation* performance of the PSLB algorithm is the average performance for the partial selection of the lower bounds using quickselect.

Table 1 The average computation time ($N_{TA} \leq N$ and $\delta_2 \leq \delta_3 \leq \delta_4 \leq \delta_1$ in most cases).

	lb calculation	lb preparation	DTW calculation
NLB	$m(N-k)$		$(k+\delta_1)n^2$
SLB	mN	$O(N \log N)$	$(k+\delta_2)n^2$
PSLB	mN	$O(N)$	$(k+\delta_3)n^2$
TALB	mN_{TA}		$(k+\delta_4)n^2$

We discuss δ_1, δ_2, δ_3, and δ_4. SLB, PSLB, and TALB follow the assumption that the DTW distance between a candidate and the query should be small if its lower bound is small. Thus, the magnitude relations of deltas are not theoretically guaranteed. However, when we perform experiments on synthetic data (details of the generation process are described in Sect. 5.1.1) in the case where $m = 5$, $N = 1{,}000$, $n = 16$ and $k = 10$, we obtain $\delta_1 = 245$, $\delta_2 = 140$, $\delta_3 = 142$, $\delta_4 = 169$, which satisfies the inequality $\delta_2 \leq \delta_3 \leq \delta_4 \leq \delta_1$.

Furthermore, we describe the space complexity. The following discussion is common to all the proposed algorithms. When loading all data structures calculated by the PREPROCESS PHASE into memory, the space complexity required to preserve them is $O(mN^2)$. Moreover, when loading only the corresponding data structure of one search query into memory, the space complexity required to preserve it is $O(mN)$. In our experiments, we consider loading only the corresponding data structure of the search queries utilized for the similar search into memory. In addition, the space complexity required for the DTW calculation is $O(n^2)$.

5. Experiments

5.1 Experimental Setting

In this section, we evaluate the effectiveness of the proposed lower bound using a synthetic dataset. In addition, we compare the performance of the DTW calculation without the lower bound (BASE) and that of the four proposed algo-

rithms (NLB, SLB, PSLB, and TALB) using two types of datasets: a real soccer dataset and a synthetic dataset. Sports must be analyzed from various perspectives; therefore, a sports dataset is a suitable means for testing our proposed flexible similarity search framework. Increasingly detailed soccer data are being collected using current technology, and analyzing these data is important for coaches, clubs and players [3]. First, we provide brief overview of two datasets; then, we present the time-dependent features of each dataset in detail and the measures of the distance between spatial points and between time-dependent feature values.

5.1.1 Datasets

In this experiment, we consider soccer player tracking data from Data Stadium Inc.[†]. This dataset consists of tracking data from 6 games observed 25 times per second. In the soccer player tracking data, each spatial point can be represented by a pair of $x^i(j)$ and $y^i(j)$ rather than by $p^i(j)$. In this experiment, we extract the FW (a forward in soccer) trajectories in the penalty area and the concurrent MF (a midfielder in soccer) trajectories from the dataset. We consider the FW trajectories to be associated with two intrinsic features, namely, the FW's speed and direction, and two extrinsic features, namely, the distance between the FW and the MF and the direction from the FW to the MF. The definitions of speed and direction are discussed in greater detail later. In summary, we consider enriched trajectories whose elements are as follows: ID, timestamp, $x^i(j)$, $y^i(j)$, FW speed, FW direction, distance between FW and MF, and direction from FW to MF. We utilize 9736 trajectories with an average length of 477.

In addition, we consider synthetic data whose elements are as follows: ID, timestamp, $x^i(j)$, $y^i(j)$, speed, direction and the two extrinsic features. To prepare these data, $x^i(j)$, $y^i(j)$ and the two extrinsic features were generated using the following random walk model in accordance with [15]:

$$v(i) = v(i-1) + dif(i)$$

where $v(1)$ and $dif(i)$ are uniformly distributed in the range $(0, 20)$.

5.1.2 Speed and Direction

We denote a speed by $fv_1^i(j)$ and a direction by $fv_2^i(j)$. The speed $fv_1^i(j)$ can be defined as the distance between adjacent points if the points are sampled at equal time intervals:

$$fv_1^i(j) = \sqrt{(x^i(j+1) - x^i(j))^2 + (y^i(j+1) - y^i(j))^2}$$

Next, we define $fv_2^i(j)$ as follows:

$$fv_2^i(j) = \begin{cases} \theta & (x^i(j+1) > x^i(j)) \\ \theta + \pi & (x^i(j+1) < x^i(j),\ y^i(j+1) \geq y^i(j)) \\ \theta - \pi & (x^i(j+1) < x^i(j),\ y^i(j+1) < y^i(j)) \\ \frac{\pi}{2} & (x^i(j+1) = x^i(j),\ y^i(j+1) > y^i(j)) \\ -\frac{\pi}{2} & (x^i(j+1) = x^i(j),\ y^i(j+1) < y^i(j)) \end{cases}$$

where

$$\theta = \arctan\left(\frac{y^i(j+1) - y^i(j)}{x^i(j+1) - x^i(j)}\right) \quad \left(-\frac{\pi}{2} < \arctan(x) < \frac{\pi}{2}\right)$$

In this formulation, a direction of pure east has an angle of 0, and a direction of pure north has an angle of $+\frac{\pi}{2}$.

Note that we cannot define the speed and direction observed at the end of a trajectory $(fv_1^i(n), fv_2^i(n))$; therefore, we do not consider the ends of the enriched trajectories. In addition, in the case of an unmoving object, we set the speed and direction to 0.

5.1.3 Distance between Spatial Points

Here, we define the distance between two spatial points. The Euclidean distance between spatial points has a larger range than the distance between time-dependent feature values (as defined below); therefore, we divide the Euclidean distance by $\sqrt{2}$.

$$D_p(p^i(j), p^g(h)) = \sqrt{\frac{(x^g(h) - x^i(j))^2 + (y^g(h) - y^i(j))^2}{2}}$$

Stricter adjustment of the distance scales will be a task for future work.

5.1.4 Distance between Time-Dependent Feature Values

The distance between four time-dependent feature values is the absolute value of the difference between the time-dependent feature values.

$$D_{fv_l}(fv_l^i(j), fv_l^g(h)) = |fv_l^i(j) - fv_l^g(h)|$$

5.1.5 Normalization

In this experiment, we consider various types of time-dependent feature values whose ranges are distinct. For instance, the range of speeds is theoretically $[0, \infty)$, whereas the range of directions is $[-\pi, \pi]$. To obtain meaningful results, we normalize $x^i(j)$, $y^i(j)$, and each time-dependent feature value such that they follow a distribution with an average of 0 and a variance of 1.

5.2 Experimental Results

We implemented the proposed algorithms in C++ and compiled them using g++ 5.2.1. The experiments were run under Ubuntu 14.04.3 on a computer with an Intel Core i7 CPU at 4.00 GHz and 32 GB of RAM. In addition, we

[†]https://www.datastadium.co.jp/

utilized PostgreSQL 9.4.7 to store the data structure computed in the PREPROCESS PHASE. We loaded all trajectory data and the data structure of the search queries utilized in the SEARCH PHASE into the main memory prior to the SEARCH PHASE, and then we measured the time taken by the SEARCH PHASE. The executable code is available at [25].

5.2.1 Comparison of the Lower Bounding Measures

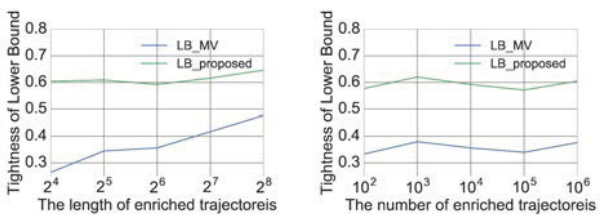
We compare the performances of the proposed lower bound and LB_MV [17], which is the extended LB_Keogh [14] for multivariate time series. We evaluate their performances using four time-dependent feature sequences on a random-walk dataset. To compare between our proposed lower bound and LB_MV, we use the Sakoe-Chiba band [26], which is a global path constraint for DTW, and the squared difference as a measure of the distance between time-dependent feature values. In addition, we utilize the *tightness of the lower bound*, which is defined as the ratio of the lower bound over the true distance, as a measurement of efficiency.

$$Tightness = \frac{LowerBound}{TrueDTWDistance}$$

The *tightness of the lower bound* is a very meaningful measure [13]. The results that are obtained when we set all weights to 1 and the number of trajectories to 10,000 are shown in Fig. 4 (a). Moreover, the results that are obtained when we set the length of trajectories to 64 are presented in Fig. 4 (b). These results suggest that our proposed lower bound is considerably superior to LB_MV. In addition, note that our proposed lower bound is simply computed based on the linear combination of each distance with weights, although it needs preparation. Therefore, the cost of calculating our proposed lower bound is considerably lower than that of calculating LB_MV, whose computational cost is mn.

5.2.2 Tests on Soccer Data

We measured the time required for a top-10 similarity search for soccer enriched trajectories as described in Sect. 5.1.1. We utilized the 21 enriched trajectories corresponding to shooting FWs as queries. In addition, we set all the weights that are uniformly distributed in the range (0,1), compute



(a) Tightness when the number of trajectories is 10,000. (b) Tightness when the length of trajectories is 64.

Fig. 4 Tightness of Lower Bound (weights = {1,1,1,1,1}).

100 times, and take the average of all the calculation times. The time for BASE was 118.103 seconds, that for the NLB algorithm was 3.955 seconds, that for the SLB algorithm was 2.075 seconds, that for the PSLB algorithm was 2.078 seconds and that for the TALB algorithm was 2.566. This experimental result indicates that the four algorithms with our proposed lower bound achieve considerably faster searches on real data. The computational times for each algorithm are discussed in greater detail below with regard to the output efficiencies observed in the subsequent studies on synthetic data.

5.2.3 Tests on Random-Walk Data

We measured the average computing times required for similarity searches on random-walk data under various conditions using 5 enriched trajectories as queries.

The experimental results obtained when we set all weights that are uniformly distributed in the range $(0, 1)$, compute 100 times, and take the average of all the calculation times are presented in Tables 2, 3 and 4. In this experiment, the PSLB algorithm is slightly superior to the

Table 2 Average calculation times of the top 1 results (with weights that are uniformly distributed in the range $(0, 1)$).

N	n	Methods				
		BASE(S)	NLB(s)	SLB(s)	PSLB(s)	TALB(s)
1,000	16	0.048	0.005	0.005	0.004	0.004
	32	0.162	0.029	0.022	0.021	0.022
	64	0.752	0.192	0.155	0.154	0.156
	128	2.844	0.896	0.763	0.762	0.772
	256	10.236	3.284	2.729	2.730	2.764
10,000	16	0.412	0.027	0.042	0.023	0.015
	32	1.870	0.179	0.146	0.127	0.127
	64	7.093	1.013	0.792	0.774	0.783
	128	25.447	4.548	3.953	3.936	3.965
100,000	16	3.912	0.200	0.416	0.202	0.111
	32	18.047	1.014	1.021	0.812	0.821
	64	71.871	7.793	6.939	6.723	6.749
1,000,000	16	42.504	2.179	4.612	2.245	1.101
	32	190.281	6.689	7.830	5.474	5.045
10,000,000	16	425.642	23.707	78.868	32.058	7.418

Table 3 Average calculation times of the top 5 results (with weights that are uniformly distributed in the range $(0, 1)$).

N	n	Methods				
		BASE(S)	NLB(s)	SLB(s)	PSLB(s)	TALB(s)
1,000	16	0.048	0.010	0.008	0.007	0.007
	32	0.162	0.048	0.035	0.034	0.036
	64	0.756	0.281	0.219	0.218	0.226
	128	2.842	1.192	0.991	0.992	1.016
	256	10.233	4.551	3.765	3.781	3.840
10,000	16	0.412	0.042	0.049	0.030	0.027
	32	1.874	0.302	0.245	0.227	0.234
	64	7.096	1.507	1.210	1.192	1.219
	128	25.445	6.485	5.395	5.381	5.452
100,000	16	3.897	0.239	0.436	0.222	0.174
	32	18.065	1.647	1.480	1.270	1.337
	64	71.890	11.025	9.416	9.200	9.297
1,000,000	16	42.486	2.383	4.708	2.343	1.549
	32	190.608	10.265	10.456	8.103	8.086
10,000,000	16	426.410	24.206	73.087	30.183	11.055

Table 4 Average calculation times of the top 10 results (with weights that are uniformly distributed in the range $(0, 1)$).

N	n	Methods				
		BASE(S)	NLB(s)	SLB(s)	PSLB(s)	TALB(s)
1,000	16	0.050	0.014	0.011	0.009	0.011
	32	0.167	0.061	0.044	0.043	0.047
	64	0.747	0.324	0.254	0.252	0.264
	128	2.882	1.352	1.110	1.113	1.152
	256	10.138	5.098	4.226	4.246	4.339
10,000	16	0.410	0.052	0.055	0.036	0.035
	32	1.856	0.360	0.290	0.272	0.284
	64	7.029	1.745	1.382	1.364	1.401
	128	25.181	7.365	6.128	6.121	6.205
100,000	16	3.879	0.269	0.463	0.242	0.207
	32	17.927	1.978	1.715	1.504	1.613
	64	71.283	12.435	10.457	10.238	10.401
1,000,000	16	42.375	2.529	5.582	2.978	1.991
	32	188.863	11.982	12.243	9.712	9.699
10,000,000	16	423.243	24.580	103.543	41.028	13.424

Table 5 Average calculation times of the top 1, 5, and 10 results (with weights = $\{0, 0, 0, 0, 1\}$).

		Top 1		Top 5		Top 10	
N	n	Methods		Methods		Methods	
		PSLB(s)	TALB(s)	PSLB(s)	TALB(s)	PSLB(s)	TALB(s)
1,000	16	0.0015	0.0002	0.0019	0.0007	0.0021	0.0013
	32	0.0017	0.0004	0.0027	0.0019	0.0035	0.0039
	64	0.0024	0.0018	0.0058	0.0099	0.0101	0.018
	128	0.0038	0.0073	0.0171	0.0358	0.029	0.0698
	256	0.0146	0.0223	0.0492	0.1161	0.101	0.2161
10,000	16	0.0182	0.0001	0.0185	0.0007	0.0189	0.0011
	32	0.0185	0.0005	0.0196	0.0023	0.0218	0.0048
	64	0.019	0.0015	0.0236	0.0085	0.0283	0.0169
	128	0.0225	0.0059	0.0331	0.0278	0.0474	0.0533
100,000	16	0.1894	0.0001	0.1903	0.0006	0.1906	0.0013
	32	0.1923	0.0005	0.1933	0.0024	0.197	0.0045
	64	0.2747	0.002	0.2758	0.009	0.2856	0.0168
1,000,000	16	2.171	0.0002	2.1949	0.0007	2.183	0.0013
	32	2.2063	0.0006	2.2195	0.0027	2.3085	0.0052
10,000,000	16	25.6581	0.0002	26.3536	0.0008	25.7757	0.0015

Table 6 Standard deviation of the calculation times of the top 10 results (with weights that are uniformly distributed in the range $(0, 1)$).

N	n	Methods				
		BASE	NLB	SLB	PSLB	TALB
1,000	16	0.004	0.005	0.004	0.004	0.005
	32	0.013	0.021	0.020	0.020	0.020
	64	0.074	0.142	0.145	0.145	0.143
	128	0.234	0.507	0.533	0.535	0.522
	256	1.272	2.021	2.178	2.188	2.144
10,000	16	0.033	0.017	0.011	0.011	0.015
	32	0.134	0.175	0.150	0.150	0.154
	64	0.790	1.007	0.942	0.942	0.945
	128	2.954	3.964	3.859	3.853	3.863
100,000	16	0.374	0.070	0.060	0.050	0.092
	32	0.856	1.228	0.991	0.989	1.057
	64	10.717	9.653	8.934	8.927	9.029
1,000,000	16	5.673	0.525	0.731	0.537	1.094
	32	8.367	10.568	8.784	8.703	9.518
10,000,000	16	72.275	1.486	22.818	9.445	8.196

Table 7 Relative standard deviation of the calculation times of the top 10 results (with weights that are uniformly distributed in the range $(0, 1)$).

N	n	Methods				
		BASE	NLB	SLB	PSLB	TALB
1,000	16	0.072	0.353	0.394	0.465	0.437
	32	0.078	0.349	0.446	0.462	0.423
	64	0.099	0.438	0.573	0.575	0.543
	128	0.081	0.375	0.480	0.481	0.453
	256	0.125	0.396	0.515	0.515	0.494
10,000	16	0.082	0.338	0.210	0.318	0.440
	32	0.072	0.488	0.518	0.552	0.543
	64	0.112	0.577	0.682	0.690	0.674
	128	0.117	0.538	0.630	0.630	0.623
100,000	16	0.096	0.260	0.130	0.208	0.444
	32	0.048	0.621	0.578	0.658	0.655
	64	0.150	0.776	0.854	0.872	0.868
1,000,000	16	0.134	0.208	0.131	0.180	0.550
	32	0.044	0.882	0.717	0.896	0.981
10,000,000	16	0.171	0.060	0.220	0.230	0.611

other algorithms in most cases. However, when the number of enriched trajectories is large and the average length of the enriched trajectories is short, then the TALB algorithm outperforms the PSLB algorithm. When the number of enriched trajectories is 10,000,000 and the average length of the enriched trajectories is 16, then the TALB algorithm is considerably superior to the PSLB algorithm. This is because the PSLB algorithm needs preparation time for the partial selection, which requires an average complexity of N, whereas the TALB algorithm does not require any preparation, as shown in Table 1.

The TALB algorithm is far superior to the PSLB algorithm when one time-dependent feature is assigned a larger weight than the others. This is because the enriched trajectories for which the DTW distances of the time-dependent features are small are more likely to be output in this case. Thus, the TALB algorithm, which accesses the lists sorted by each distance, performs well. The results obtained in the case where we set one weight to 1 and the remaining weights to 0, in which TALB works best, are shown in Table 5. Although the PSLB algorithm performs slightly better under certain conditions, we consider that these experimental results suggest that the TALB algorithm performs well under any condition.

In addition, we show the standard deviation and the relative standard deviation, which is the ratio of the standard deviation to the average, of the 100 calculation times of the top 10 results when we set all weights that are uniformly distributed in the range $(0, 1)$ in Table 6 and in Table 7. Moreover, boxplots of the 100 calculation times of the top 10 results using one query when we set all weights that are uniformly distributed in the range $(0, 1)$, N to 10,000 and n to 128 are shown in Fig. 5, and boxplots when we set N to 1,000,000 and n to 16 are shown in Fig. 6. Table 7 shows that the dispersion of the calculation times of TALB is larger than that of other algorithms when the average length of trajectories is short and the number of trajectories is large. However, Figs. 5 and 6 suggest that the best computation time of TALB can be superior to that of other alogrithms and the worst computation time of TALB is not inferior to that of other algorithms. This result is because when one time-dependent feature is assigned a larger weight than others, the calculation time of TALB is close to 0 in any case as described above. Thus, the TALB algorithm

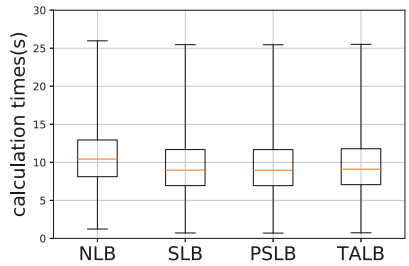


Fig. 5 Boxplots of the calculation times of the top 10 results when we set all weights to the values that are uniformly distributed in the range $(0, 1)$, N to 10,000 and n to 128.

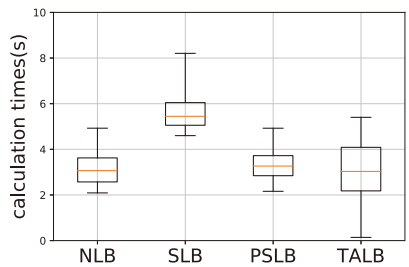


Fig. 6 Boxplots of the calculation times of the top 10 results when we set all weights to the values that are uniformly distributed in the range $(0, 1)$, N to 1,000,000 and n to 16.

is suitable for practical use.

6. Conclusions and Future Work

We proposed a framework for flexible similarity searches for enriched trajectories. Moreover, we proposed a lower bounding measure and four algorithms based on the measure. We evaluated their performances under various conditions in our experiments.

There are three main prospective directions for future research, as follows:

1. estimating the search intent
2. accelerating the PREPROCESS PHASE
3. searching for sub-enriched trajectories

First, the need to input all the weights imposes a high cost on users; to alleviate this cost, we wish to impart the framework with the ability to estimate the search intent.

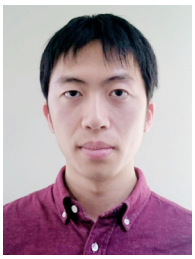
Second, we need to accelerate the PREPROCESS PHASE. Let m be the number of time-dependent features, let n be the average length of the enriched trajectories, and let N be the number of enriched trajectories. Then, the cost of computing the DTW distances for all features between all pairs of enriched trajectories is mn^2N^2. This result indicates that the calculation costs substantially increase as the average length and number of enriched trajectories increase. Therefore, it will be important to improve the efficiency of the PREPROCESS PHASE for practical use.

Finally, our proposed framework can identify similar overall enriched trajectories but might miss sub-enriched trajectories. In some cases, it is impossible to obtain the desired insight unless sub-enriched trajectories are considered. Therefore, it will be desirable to enhance the capability of our framework to address sub-enriched trajectories.

References

[1] J.-G. Lee, J. Han, and K.-Y. Whang, "Trajectory clustering: A partition-and-group framework," Proc. ACM SIGMOD International Conference on Management of Data, pp.593–604, 2007.
[2] B. Ristic, B. La Scala, M. Morelande, and N. Gordon, "Statistical analysis of motion patterns in ais data: Anomaly detection and motion prediction," Proc. International Conference on Information Fusion, pp.1–7, 2008.
[3] J. Gudmundsson and T. Wolle, "Towards automated football analysis: Algorithms and data structures," Proc. Australasian Conference on Mathematics and Computers in Sport, pp.145–152, 2010.
[4] C. Yajima, Y. Nakanishi, and K. Tanaka, "Querying video data by spatio-temporal relationships of moving object traces," Proc. IFIP TC2/WG2.6 Sixth Working Conference on Visual Database Systems, pp.357–371, 2002.
[5] J.-G. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partition-and-detect framework," Proc. IEEE International Conference on Data Engineering, pp.140–149, 2008.
[6] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," Proc. IEEE International Conference on Data Engineering, pp.673–684, 2002.
[7] X. Gong, Y. Xiong, W. Huang, L. Chen, Q. Lu, and Y. Hu, "Fast similarity search of multi-dimensional time series via segment rotation," Proc. International Conference on Database Systems for Advanced Applications, pp.108–124, 2015.
[8] N. Pelekis, I. Kopanakis, G. Marketos, I. Ntoutsi, G. Andrienko, and Y. Theodoridis, "Similarity search in trajectory databases," Proc. International Symposium on Temporal Representation and Reasoning, pp.129–140, 2007.
[9] M. Buchin, A. Driemel, M. van Kreveld, and V. Sacristán, "An algorithmic framework for segmenting trajectories based on spatio-temporal criteria," Proc. ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp.202–211, 2010.
[10] K. Zheng, S. Shang, N.J. Yuan, and Y. Yang, "Towards efficient search for activity trajectories," Proc. IEEE International Conference on Data Engineering, pp.230–241, 2013.
[11] Y. Zheng, F. Liu, and H.-P. Hsieh, "U-Air: When urban air quality inference meets big data," Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.1436–1444, 2013.
[12] D.J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series.," Proc. ACM KDD Workshop, pp.359–370, Seattle, WA, 1994.
[13] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," Data Mining and Knowledge Discovery, vol.26, no.2, pp.275–309, 2013.
[14] E. Keogh and C.A. Ratanamahatana, "Exact indexing of dynamic time warping," Knowledge and Information Systems, vol.7, no.3, pp.358–386, 2005.
[15] Y. Sakurai, M. Yoshikawa, and C. Faloutsos, "FTW: Fast similarity search under the time warping distance," Proc. ACM Symposium on Principles of Database Systems, pp.326–337, 2005.
[16] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," Journal of Computer and System Sciences, vol.66, no.4, pp.614–656, 2003.
[17] T.M. Rath and R. Manmatha, "Lower-bounding of dynamic time warping distances for multivariate time series," Tech Report MM-40, University of Massachusetts Amherst, 2003.
[18] Y. Zheng, "Trajectory data mining: An overview," ACM Trans. Intelligent Systems and Technology, vol.6, no.3, Article No.29, 2015.

[19] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.262–270, 2012.
[20] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," Intelligent Data Analysis, vol.11, no.5, pp.561–580, 2007.
[21] W. Yu and M. Gertz, "Constraint-based learning of distance functions for object trajectories," Proc. International Conference on Scientific and Statistical Database Management, pp.627–645, 2009.
[22] M. Reyes, G. Domínguez, and S. Escalera, "Featureweighting in dynamic timewarping for gesture recognition in depth data," Proc. IEEE ICCV Workshop, pp.1182–1188, 2011.
[23] C.A.R. Hoare, "Partition:algorithm 63, Quicksort:algorithm 64, Find:algorithm 65," Commun. ACM, vol.4, no.7, pp.321–322, 1961.
[24] H. Bast, D. Majumdar, R. Schenkel, M. Theobald, and G. Weikum, "IO-Top-k: Index-access optimized top-k query processing," Proc. International Conference on Very Large Data Bases, pp.475–486, 2006.
[25] "Source code used in the experiment." https://github.com/eimen/Enriched_trajectory_search/
[26] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," IEEE Trans. Acoust. Speech Signal Process., vol.26, no.1, pp.43–49, 1978.



Hideaki Ohashi received his B.E. degree from the School of Informatics and Mathematical Science, Kyoto University, in 2015. He is currently a master course student at the Graduate School of Informatics, Kyoto University. His research interests include time-series data mining and trajectory data mining. He is a member of DBSJ.

Toshiyuki Shimizu received his B.E. degree in 2003, M.S. degree in Information Science from Nagoya University in 2005, and Ph.D. degree in Informatics from Kyoto University in 2008. He is currently an assistant professor at the Graduate School of Informatics, Kyoto University. His research interests include handling semi-structured data, science data management, and scholarly databases. He is a member of ACM, IPSJ, and DBSJ.

Masatoshi Yoshikawa received his B.E., M.E., and Dr. Eng. degrees from the Department of Information Science, Kyoto University, in 1980, 1982, and 1985, respectively. From 1985 to 1993, he was with Kyoto Sangyo University. In 1993, he joined the Nara Institute of Science and Technology as an associate professor in the Graduate School of Information Science. From June 2002 to March 2006, he served as a professor at Nagoya University. Since April 2006, he has been a professor at Kyoto University. His general research interests are in the area of databases. His current research interests include multi-user routing algorithms and services, theory and practice of privacy protection, and medical data mining. He is a member of the ACM and IPSJ.