Similarity Search in Moving Object Trajectories

Omnia Ossama Hoda M. O. Mokhtar

Information Systems Dept.
Faculty of Computers and Information
Cairo University
omnia@ieee.org hmokhtar@gmail.com

Abstract

The continuous and rapid advent in mobile and communications technology opens the way for new research areas and new applications. Moving Object Databases(MODs) are among the emerging research topics that are attracting many work due to their vital need in many applications. Generally, MODs deal with geometries changing over time. In this paper we study an interesting point in moving object databases; namely, similarity search between moving object trajectories. We propose a novel similarity search technique that is based on trajectory alignment using gaps. Our approach binds the idea of biological sequence alignment using the Needleman-Wunsch algorithm [3] together with different similarity search distances. Finally, we present an empirical study for the efficiency and accuracy of our proposed approaches.

1 Introduction

With the development of wireless communications and positioning technologies, the concept of moving objects databases (MOD) became increasingly important and is currently applied in numerous daily applications. The recent advances in mobile computing and global positioning systems (GPSs) have made it possible to collect large amounts of spatio-temporal data that need to be efficiently managed and queried. Many emerging applications therefore present an increasing demand to perform efficient data analysis tasks over these data. For example, traffic analysis applications, m-commerce, E-911, weather forecast, animal migration analysis, along with a wide range of location based services (LBSs) as locating nearest service, fleet management, and find my friend application [14] are all applications that require storing, managing, and querying moving objects. In addition, analysts forecast that LBS revenue will reach an annual global total of $13.3 billion by 2013, up from an estimated $515 million during 2007 [1].

15th International Conference on Management of Data
COMAD 2009, Mysore, India, December 9–12, 2009

Inspired by the importance of trajectory similarity area, in this paper we present possible techniques that efficiently help in ranking moving objects according to their trajectory similarity. Our proposed similarity search approach can be applied in many real world applications like air traffic management and control problems where collision detection and prevention are crucial tasks to avoid accidents among aircrafts that have to follow same routes [8]. Road network traffic management and accident prediction is another possible application.

The remainder of the paper is organized as follows: Section2 presents a literature survey of key related work. Section 3 discusses moving object trajectories. Section 4 provides a comparative analysis between different similarity search techniques for moving object trajectories and presents our proposed similarity search techniques. Experimental evaluation is provided in section 5. Finally, section 6 concludes the paper and proposes some challenging future work.

2 Related Work

For a long time similarity search in time series data has been a focus in many research work. Similarity search has been well studied in the time series analysis domain as presented in [13] [5] [10] [9]. The earliest and basic approach for fast similarity search is the Euclidean distance measure. Euclidean distance approach is considered the original approach for defining the similarity between two trajectories through viewing them as vectors and using Euclidean distance between them to calculate the similarity distance [2]. The weaknesses of Euclidean distance in handling noise and local time shifting have motivated the need for new similarity measures. In [13], the authors introduce the concept of Dynamic Time Warping (DTW). The main idea behind the DTW technique is to allow a time series to be stretched to provide better match with another series. This approach enabled comparing sequences with different lengths which in turn appeared to be useful in comparing moving object trajectories with different lengths as well. The algorithm of the DTW technique finds the optimal match between two trajectories which may vary in time or speed. The two trajectories are warped to deter-

mine a measure, namely, the *warping path* of their similarity. Although dynamic time warping incurred a heavy computation cost, nevertheless it was more robust against noise than the Euclidean distance approach.

In [5], the authors proposed the ERP distance function for Edit distance with Real Penalty. The ERP distance combines the L1-norm [2] and the edit distance [6]. ERP supports time shifting, and it is a metric function satisfying triangle inequality. They developed pruning strategies for large time series databases through applying either triangle inequality or using a lower bound on the ERP distance. Moreover, the authors showed that these two ways of pruning can be used simultaneously for the ERP distance. Inspired by the importance of the similarity problem in moving object databases (MODs), the authors in [11] studied the time independent similarity search problem of moving object trajectories. The problem was though presented from a different perspective, trajectories were compared based only on their shapes. The authors proposed the One Way Distance (OWD) function that compares the *spatial shapes* of trajectories along with appropriate algorithms for computing the one way distance. The basic features of this technique are: it is not time sensitive, lengths of trajectories can be different, and finally, the mapping between the trajectories may not be continuous. The main problem of this technique is ignoring the time dimension. Moreover, in [6] the authors proposed the use of Edit Distance on Real sequence (EDR) to trajectory data retrieval and presented a dimensionality reduction technique through a symbolic representation of trajectories. Finally, in [7] the authors used the Fréchet distance as a proximity measure to define distance between time series however they focused on the concept of time series clustering and pattern recognition tasks.

3 Moving Object Databases

In the past few years, we have witnessed the birth of a new database area, namely, moving object databases (MODs). Moving object databases attract the attention of many researchers due to their continuous contribution in many mobile and location based applications. Simply, moving objects are defined as objects whose time and/or extend change over time [15]. In this paper, we focus on *moving points* where extent is ignored and only location changes are maintained in the database. For simplicity, we consider 2D moving points like cars, buses, trains, mobile users, migrating birds, etc. We treat time T as a separate continuous dimension over real numbers. We also use the notion of time intervals which are basically real intervals [12]. The main characteristic of moving objects is their trajectory, which simply defines their position changes over time. More precisely, a trajectory is the path a moving point follows along its route. It is used to represent the dynamic attributes of the moving object, including the past, current, and future location information [15]. The location of a moving point can be modeled by a linear time-parameterized function of the form $x = At + B$ where A, B are reals. In our model we define a trajectory as a sequence of continuous line segments (i.e. *motions*). In other words, a trajectory is piece-wise linear. A *motion* is a segment of the trajectory such that at each time instant during the object's movement, its position is defined by a point on a motion.

Definition 1 *A motion m is a line segment defined as a time-parameterized linear function of time of the form $At+B$. A motion is defined by the pair (A, B). Where, A, B are real numbers, A is the segment's slope, and B is the intercept.*

Using the slope and intercept pair, each motion is fully defined. Therefore, the position of the object along a motion at any time instant can be recorded.

Definition 2 *Let n be a natural number such that $n > 1$. A moving object trajectory is a sequence $(t_0, m_0, t_1, m_1, \cdots, t_{n-1}, m_n, t_n)$. Such that for all $1 \leqslant i \leqslant n$ the following 2 conditions should be satisfied. The first is time continuity: $t_i \in T$ is a time instant such that for each $1 \leqslant i \leqslant n$, $t_i < t_{i+1}$. The second is motion continuity: m_i is a motion representing the path of the object during the time period $[t_i, t_{i+1}]$ such that $m_i(t_{i+1}) = m_{i+1}(t_{i+1})$.*

4 Similarity Search in Moving Object Databases

Similarity search is a crucial and fundamental task in data mining. 1D time series data as stock or commodity prices, sales volume, weather data and biomedical measurements have been the focus of many similarity search techniques [2] Although the proposed techniques are interesting, their applicability to higher dimensional data is not straightforward. Moving object trajectories are examples of 3D data (including time as a dimension) that could be similar in total or in part. They may start and end together but due to the different speeds or sampling rates of different trajectories, they possess high internal dissimilarity among their motions. In general, different similarity search techniques address the similarity from different perspectives, each technique has its own similarity measure and/or distance function. Therefore, similarity search techniques are usually classified based on the type and features of the distance function they use. In this section we introduce similarity search techniques that employ the idea of sequence alignment that is applied in aligning biological data. We propose three versions for our approach, the first basically elevates the Needleman-Wunsch [3] algorithm used for global pairwise sequence alignment of biological data to be adapted for aligning moving object trajectories. We refer to this version as trajectory alignment using gaps (TAG). The second version of our approach adds an additional step. This step is required to compute the Euclidean distance between two aligned trajectories. We refer to this version as TAG-ED (i.e. trajectory alignment with gaps and Euclidean distance). The final version uses TAG and adds a distance

measure step that is based on the ERP distance in [5], and we refer to this version as TAG-ERP (i.e. trajectory alignment with gaps and ERP distance).

In biological data sequence alignment a crucial step is to compare sequences from different species. In general, sequence alignment implies finding exact matches, handling dissimilarities by adding mismatch penalties, and finally inserting gaps among sequences to obtain sequences of same length. The Needleman-Wunsch algorithm finds the optimal alignment of a pair of biological sequences by optimizing a score function, that is, each possible alignment is scored based on a score function, and the alignment that yields the highest score is considered the optimal alignment of the two sequences [16]. The algorithm allows gaps to be inserted into the sequences to enhance the alignment. However, alignments that include gaps yield a lower score than alignments that do not include gaps. In other words, introducing gaps into an alignment is penalized by assigning gap insertions a negative score term (a gap penalty).

Driven by the fact that moving object trajectories could easily have different lengths due to the use of different sampling rates, aligning trajectories before applying similarity search technique turns out to be a reasonable approach.

In [5] the authors propose a distance function to compute the similarity between two 1D time series sequences. Given 2 sequences R=$\{r_1, r_2, r_3, \ldots, ri, \ldots, r_n\}$ and S= $\{s_1, s_2, s_3, \ldots, si, \ldots, s_m\}$, where n and m are integers and could be of the same length. g is the penalty cost and most likely it will be zero [5]. The ERP distance is defined as:

$$ERP(r_i, s_i) = \begin{cases} |r_i - s_i| \; if \; r_i, s_i \; both \; are \; not \; empty \\ |r_i - g|, \; if \; s_i \; is \; empty \\ |s_i - g|, \; if \; r_i \; is \; empty \end{cases}$$

Although [5] proposes a neat idea, our main contributions are the following:

1. The ERP technique works with 1D time series data, generalization to higher dimensions is not clear. Here we consider 2D moving objects.
2. The ERP technique adds gaps to sequences to obtain sequences of same length, however the gaps are always inserted at the end of the sequence [5]. Here we allow gaps to be inserted among the trajectory based on the best alignment score from applying the Needleman-Wunsch algorithm [3].
3. The ERP technique assigns no penalty for gaps (gap penalty =0) [5]. Here we study the effect of the gap cost on alignment and similarity accuracy.
4. We propose 3 techniques that are based on our TAG approach (Trajectory Alignment using Gaps) that differ in their distance measure computation.
5. We present an empirical study that compares our techniques against other techniques.

In the next section we explore our proposed techniques and how they handle trajectory data.

4.1 A Novel Similarity Computation Approach

Our approach is based on finding an optimal alignment between the query trajectory and the trajectories in MOD. The result of this alignment is a set of equal length trajectories that can then be compared to compute their similarity measure. The main advantage in our technique is that we insert gaps into the trajectory only when needed and at the appropriate time instants. In regular biological alignment problems, a gap is simply a space inserted into the 1D data sequence. In our approach a trajectory gap is defined as shown in Definition 3.

Definition 3 *Let t_i be a time instant such that $j \leqslant i \leqslant k$. Given a trajectory τ of length n such that $\tau = (t_0, m_0, t_1, m_1, \cdots, t_j, m_k, t_k, \cdots, t_{n-1}, m_n, t_n)$. A trajectory gap at time t_i is an additional sampling point that is inserted into τ to give a new trajectory τ' such that:*
$| \tau' | = | \tau | + 1$
$\tau' = (t_0, m_0, t_1, m_1, .., t_j, m_k, t_i, m_k, t_k, .., t_{n-1}, m_n, t_n)$

Using trajectory gaps we first align the trajectories in MOD so that we obtain trajectories with same length, possibly with gaps inserted into them if required, thus: Given a set of n-trajectories $T_1, T_2, \cdots, T_n$ a *Trajectory Alignment using Gaps problem* (TAG) is the problem of finding the optimal n-trajectory set alignment $T_1', T_2', \cdots, T_i', \cdots, T_n'$, where, $| T_1' | = | T_2' | = \cdots = | T_n' |$ T_i' is an extension of T_i by insertion of trajectory gaps. Following Definition 2 of moving object trajectories. We allow trajectories to have different number of motions. Variable trajectory length is then treated through adding *trajectory gaps* at appropriate positions to obtain the best similarity result. Having a trajectory motion defined by its slope and intercept pair, our similarity search model considers four motion similarity cases: MATCH, SLOPE_MATCH, INTERCEPT_MATCH, AND MISMATCH. The first case is MATCH where both trajectories have identical motions, i.e. same slope and intercept for each motion. This implies that both trajectories follow the same motion pattern during their whole trip period. Consequently, the trajectories have same number of motions and no gaps are added. This case is thus scored the maximum similarity score. The second case is SLOPE_MATCH where both trajectories have at least 1 motion with equal slope but different intercept. This case represents the case of objects moving on parallel routes, so the routes have the same slope but they are not the exact same route. This motion pattern is common for objects along road networks. Such case is also favored as trajectories are structurally the same. Therefore, a small penalty is applied. The third case is INTERCEPT_MATCH where both trajectories have some motions with equal intercepts but different slopes. This case represents the case of objects moving on different intersecting routes. A higher penalty is thus applied as shape similarity is lost. The last case is MISMATCH where both trajectories are absolutely different. They have different slopes and different intercepts. Maximum penalty is thus applied. The four cases are presented in Figure 1. For simplicity the figure illus-

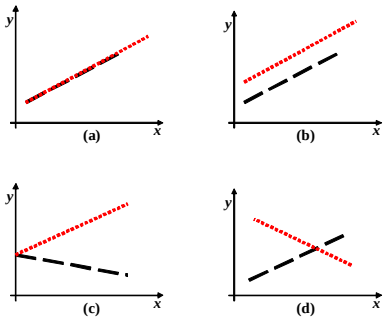

Figure 1: Trajectory Similarity Cases: (a) MATCH (b) SLOPE_MATCH (c) INTERCEPT_MATCH (d) MISMATCH

trates the cases for a single motion of a trajectory. In brief, our approach works as follows: Given a MOD D, query trajectory $Q \in D$. We iterate on all trajectories in D and compute their similarity against Q. Trajectories are then stored in a sorted list that reflects their degree of similarity with Q. Thus, given query trajectory of length k (i.e. $\mid Q \mid = k$) and a test trajectory $\tau \in D$ such that length of $\tau = l$ (i.e. $\mid \tau \mid = l$) and k is not necessarily equal to l. We place the motions of Q along one dimension of a matrix and the motions of τ along the other dimension. We then follow a similar approach to the Needleman-Wunsch algorithm [3]. We first assume values for the scores of the 4 cases above and a gap penalty. For each cell in the matrix, its value is then computed according to algorithm `Compute_Similairty`. The algorithm implements a dynamic programming approach to compute the similarity score through comparing the score of the 4 similarity cases MATCH, SLOPE_MATCH, INTERCEPT_MATCH, AND MISMATCH against the score of adding a trajectory gap into one of the 2 trajectories. The maximum score is then stored in the matrix. The right bottom cell in the matrix defines the final similarity score that is then used to sort the trajectories accordingly. Along our computations we keep track of the direction (up, left, diagonal) from which each cell in the matrix was computed. This part is required to enable us to back trace to get the aligned trajectories. Let `MAX` be a function that returns the maximum value, and `SCORE` be a function that returns the score of the similarity based on the 4 cases as discussed above. The similarity computation algorithm works as shown in Figure 2.

The function `SCORE` basically computes the score of each trajectory motion. It compares the motions of the trajectories to determine which score value should apply according to their similarity case. The function simply works as shown in Figure 3. Once the score matrix is computed, a back trace is performed using the direction information stored to obtain the final trajectory alignment as applied in the Needleman-Wunsch algorithm [3]. Another version of our technique, named (TAG-ED), applies our TAG and solves it using our Algorithm `Compute_Similairty`. However, instead of using the bottom right value in the matrix directly as the alignment score and hence the similarity measure, we employ the Euclidean distance to measure our similarity value. Simply after computing the optimal trajectory alignments, all trajectories are guaranteed to have the same length and hence Euclidean distance can be applied between all sampling points. The last version, named (TAG-ERP), also is based on solving the TAG problem and then applying a different similarity measure. In this case we employ the ERP distance to compute the distance between the trajectories. The advantage here over the ERP approach presented in [5] is that the distance function is applied over optimally aligned trajectories. To enhance performance of our algorithm shown in Figure 2, we introduce a more efficient version that it is way faster from the previous straight forward algorithm. This improvement is achieved by adding a threshold condition to stop the comparisons between two significantly mismatched trajectories in earlier stage. Thus, once the trajectories considered as mismatched beyond the threshold value, future comparisons for remaining motions are pruned. The improved algorithm proceeds as shown in Figure 2 with 2 exceptions: (1) a threshold value is input to the algorithm, (2) a new stopping condition is added to the body of the loop on lines 10,11 as shown in Figure 4.

Algorithm:`Compute_Similarity`(Q,τ)

Input: Q: Query Trajectory, MOD D, τ: a trajectory in D, k: length of Q, $\mid \tau \mid$: length of τ
Output: $M = (k+1, \mid \tau \mid +1)$: a 2D score matrix

1 Let Q be on the x-axis and τ on the y-axis;
`/* Begin Initialization */`
2 M(0,0)=0;
3 **for** $(1 \leqslant i \leqslant k)$ **do**
4 M(i,0)=gap_penalty
5 **endfor**
6 **for** $(1 \leqslant j \leqslant \mid \tau \mid)$ **do**
7 M(0,j)=gap_penalty
8 **endfor**
`/* Dynamic prog. matrix filling */`
9 **for** $(1 \leqslant i \leqslant k)$ **do**
10 **for** $(1 \leqslant j \leqslant \mid \tau \mid)$ **do**
11 M(i,j) = MAX(*M(i-1, j-1)* +SCORE(Q_i, τ_j), *M(i, j-1)-gap_penalty, M(i-1, j)-gap_penalty*)
12 **endfor**
13 **endfor**
14 **return** M$(k+1, \mid \tau \mid +1)$

Figure 2: Similarity distance computation algorithm.

5 Experimental Evaluation

In this section we investigate the strengths of our proposed similarity measures through presenting empirical results. For our experiments we used the Brinkoff generator that is commonly used for generating realistic moving objects [4]. Using the generator, we simulated 2D trajectories of vehicles on the road network in the city of San Francisco. In our experiments we evaluated different similarity search techniques along two important metrics namely, perfor-

Procedure:score(m,n)

Input: $m = (a, b)$ a motion from query trajectory, $n = (x, y)$: a motion from test trajectory
Output: similarity score value
Let val(z) be similarity score value of similarity case z;
if *(a==x && b==y)* **then**
  return val(MATCH)
else if *(a==x && b!=y)* **then**
  return val(SLOPE_MATCH)
else if *(a!=x && b==y)* **then**
  return val(INTERCEPT_MATCH)
else
  return val(MISMATCH)

Figure 3: Motion score computation.

Algorithm: Similarity_With_Threshold(Q,τ)

M(i,j) = MAX(*M(i-1, j-1)* +Q_i, τ_j, *M(i,j-1)-gap_penalty, M(i-1, j)-gap_penalty*)
if *M(i,j)*$\geqslant$ *Threshold* **then**
  break;
endif

Figure 4: Enhanced similarity distance computation.

mance and accuracy. We measure the performance in terms of processing time, and since all test data sets are relatively small in size, they fit into memory and no index structure is being considered. Our experiments were conducted using Windows XP with processor Celeron(R) 2.67MHz CPU, 512MB memory, and 160GB hard disk.

The first experiment compares the performance of Euclidean distance [2], DTW [13], ERP [5], and our proposed TAG technique. We compared the running time of each algorithm versus different number of trajectories ranging from 1000 trajectory to $10,000$ trajectories that varies in length from 100 to 1000. Figure 5 shows that the Euclidean distance algorithm (ED) outperforms other similarity search techniques due to its simple calculations. DTW shows bad processing time because it use dynamic programming concept to find optimal warping path. OWD algorithm is the worst due to its high processing requirement. On the other hand, TAG takes longer time than ERP and Euclidean distance algorithms as it requires more time to obtain the optimal trajectory alignment. Yet it outperforms both DTW and OWD techniques. . The next experiment shows a comparison between the performance of our different proposed techniques, TAG, TAG-ED, TAG-ERP. Figure 6 shows that TAG requires least processing time than TAG-ED and TAG-ERP as they require additional computational time to compute their final similarity distances. In addition we regenerated the experiment with TAG algorithm with a threshold that prunes that search space through neglecting comparisons for mismatching trajectories. Figure 7 shows how this threshold improved the running time of the algo-

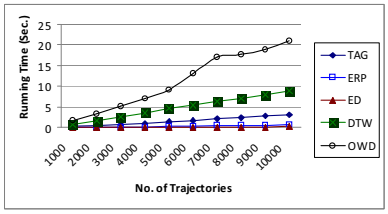

Figure 5: Performance of similarity search techniques.

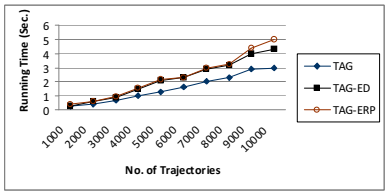

Figure 6: Performance of proposed techniques.

rithm. It takes less than a second to process 10000 trajectory compared to 4 seconds using the straight forward algorithm.

The following experiment compares the accuracy of OWD, Euclidean distance, ERP, and TAG, TAG-ED, TAG-ERP. We selected 5 trajectories from our test MOD as shown in Figure 8 to conduct our experiment on them. For this experiment we defined similarity as the minimum area between the query trajectory and the trajectories in the dataset and referred to the similarity result as the *area comparison* result.

Figure 9 shows that $T1$ which is the worst trajectory in shape and distance measure, TAG puts it in a lower rank due to its shape difference from Q, yet it favors $T2$ than $T4$ that is returned from the area ranking technique. The reason behind this result is the fact that TAG favors shape similarity. Similarly OWD puts it in a lower rank too because it addresses similarity from a shape based perspective. On the other hand $T5$ is favored by TAG-ED and TAG-ERP as both are based on distance measure and hence closer trajectories are ranked higher. Whereas it gives a high rank to $T1$ which has highly difference from Q in shape because both are based on distance measure rather than shape-based similarity. Their result are also comparable with other distance based techniques as the ERP and the Euclidean distance.

Next, we examined the effect of changing the trajectory gap penalties on the accuracy of our proposed algorithm. In this experiment we randomly selected 10 trajectories from our test MOD. We then counted the number of times our

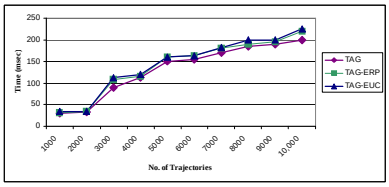

Figure 7: Performance of proposed techniques with threshold.

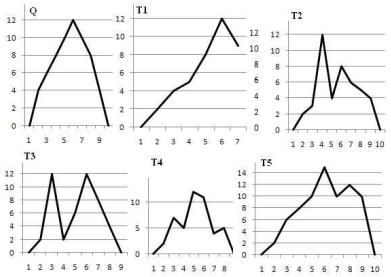

Figure 8: Sample Dataset.

Area-Ranking	Q	T4	T5	T3	T2	T1
OWD	Q	T4	T3	T2	T5	T1
TAG	Q	T2	T5	T3	T4	T1
TAG-ED	Q	T5	T1	T4	T3	T2
TAG-ERP	Q	T5	T1	T4	T3	T2
Euclidean	Q	T1	T2	T5	T3	T4
ERP	Q	T4	T5	T2	T1	T3

Figure 9: Accuracy of similarity search techniques.

proposed algorithm returns the most similar trajectory. Figure 10 shows that if we neglect the gap penalty cost (Gap penalty=0) we obtain high percentage of error. Whereas the increase in the trajectory gap penalty decreases the percentage of error. The intuition behind this result is that introducing gaps without penalty can cause dissimilar trajectories to be scored higher than more similar ones. This in turn increases the percentage of error in query result. Finally, we studied the number of occurrences of each case

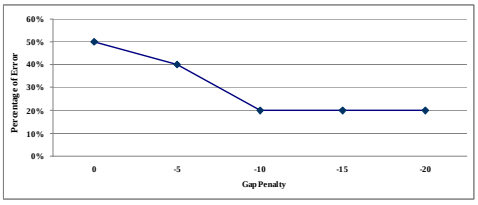

Figure 10: Effect of trajectory gap penalties.

in the similarity model, in our dataset to justify that no other cases occur and that they cover all possible relationships among motions. We did this experiment on a sample dataset of $60,000$ motions. Our result was as follows: MATCH occurred 2270 times, SLOPE_MATCH 5219 times, INTERCEPT_MATCH 8535 times, and MISMATCH 43976 times.

6 Conclusions

In this paper we presented new similarity search techniques that are based on optimal trajectory alignment using gaps. The main advantages of our techniques are: gaps are inserted into the trajectory only when needed; 2D moving objects are considered; and empirical study that compares our proposed techniques against other similarity search techniques are presented. For future work, we believe that further improvements in similarity search measures are still applicable. We also think that applying data mining techniques is another possible future direction.

References

[1] http://www.abiresearch.com/press/1097-Mobile+Location+Based+Services+Revenue+to+Reach+$13.3+Billion+Worldwide+by+2013, April 2008.

[2] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. pages 69–84. Springer Verlag, 1993.

[3] N. S. B. and W. C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, March 1970.

[4] T. Brinkhoff. Generating traffic data. *Bulletin of the Technical Committee on Data Engineering, IEEE Computer Society*, 26:19–25, 2003.

[5] L. Chen and R. Ng. On the marriage of lp-norms and edit distance. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 792–803, Toronto, Canada, 2004.

[6] L. Chen, M. T. Özsu, and V. Oria. Robust and fast similarity search for moving object trajectories. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502, Baltimore, Maryland, 2005.

[7] A. Chouakria-Douzal and P. N. Nagabhushan. Improved frchet distance for time series. In *Book Series Studies in Classification, Data Analysis, and Knowledge Organization*, pages 13–20, Berlin, Heidelberg, 2009.

[8] A. In, Y. jen Chiang, J. T. Klosowski, C. Lee, and J. S. B. Mitchell. Geometric algorithms for conflict detection/resolution in air traffic management. In *36th IEEE Conference on Decision and Control*, pages 1835–1840, 1997.

[9] E. Keogh and M. Pazzani. Derivative dynamic time warping. *First SIAM International Conference on Data Mining*, 2001.

[10] A. W.-C. F. Kin-Pong Chan. Efficient time series matching by wavelets. In *ICDE '99: Proceedings of the 15th International Conference on Data Engineering*, pages 126–133, 1999.

[11] B. Lin and J. Su. Shapes based trajectory queries for moving objects. In *GIS '05: Proceedings of the 13th annual ACM international workshop on Geographic information systems*, pages 21–30, Bremen, Germany, 2005.

[12] H. Mokhtar, J. Su, and O. Ibarra. On moving object queries: (extended abstract). In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 188–198, Madison, Wisconsin, 2002.

[13] S. Salvador and P. Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[14] J. H. Schiller and A. Voisard. *Location-based Services*. Elsevier, 2004.

[15] O. Wolfson, B. Xu, S. Chamberlain, and L. Jiang. Moving objects databases: issues and solutions. In *Proceedings of the 10th International Conference on Scientific and Statistical Database Management (SSDBM98)*, pages 111–122, Capri, Italy, July 1998.

[16] M. J. Zvelebil and J. O.Baum. *Understanding Bioinformatics*. Garland Science, Taylor & Francis Inc, 2007.