SimTraj: An Approach to Similar Queries over Trajectories in Metric Spaces

Fábio Afonso[1], Fernanda Barbosa[2], Armanda Rodrigues[1]

[1]CITI / Departamento de Informática
Faculdade de Ciências e Tecnologia da UNL
2829-516 Caparica, Portugal
fabio.afonso@gmail.com
arodrigues@di.fct.unl.pt

[2]Departamento de Informática
Faculdade de Ciências e Tecnologia da UNL
2829-516 Caparica, Portugal
fb@di.fct.unl.pt

1. Introduction

With the rapid increase in the use of location-acquisition technologies (GPS, GSM networks, etc.), large amounts of spatio-temporal datasets will be accumulated. In different application domains, we need to represent moving entities, i.e. collect the successive location positions of a given entity, which form the trajectory for that entity. The key idea is that moving entities are described by a sequence of positions in a k-dimensional space. Each position in the sequence represents the entity's location at a given time. Thus, a trajectory for a moving entity in a k-dimensional space is viewed as a line in a k+1-dimensional space, where time is an additional dimension.

In many applications, there is a need to analyze the dynamics of moving objects in order to support spatiotemporal decisions. A significant type of query in these applications is the k-nearest neighbours (kNN), which finds the k trajectories more similar (closest) to a given trajectory. For example, in a football match, to identify a player with the most similar trajectory to Ronaldo's, in order to substitute a player, without changing team's strategy; or in a tourist guide application, to find the k most similar bus trajectories to a given touristic route; or to search for the k most similar hurricanes trajectories to Katrina's.

The choice of a distance function, which best represents the degree of similarity, for trajectories in a metric space, depends on the application domain in question. Some of the most used metric functions for moving objects are: the Euclidean Distance (ED), the Manhattan Distance (MD) and the Edit distance with Real Penalty (ERP) (Chen 2005).

In order to have efficient similar searching in metric spaces, several metric data structures have been proposed, which can be classified as cluster-based or pivot-based (Samet 2006; Chávez et al. 2001). Some of these metric data structures are: Recursive Lists of Clusters (RLC) (Mamede 2005; Sarmento 2010) and Metric-Tree (M-Tree) (GBDI 2009; Ciaccia et al. 1997). Both of these metric data structures are dynamic, implemented in secondary memory and seek to minimize the number of distance computations in a similarity search. The RLC is cluster-based, while the M-tree has features common to both, pivot-based and cluster-based.

The main goal of this research is to have a trajectory storage method, based on metric data structures, that speeds up the search by similarity. In the remainder of this paper we will, firstly, describe SimTraj, which is a trajectories storage method in metric spaces that provides efficient kNN searches. Then, we present the evaluation of the performance of SimTraj method in kNN searches. This evaluation involves the two metric data structures, RLC and the M-Tree, in two

metric spaces of hurricanes trajectories. The evaluation involves the two metric data structures, RLC and the M-Tree, in two metric spaces of hurricanes trajectories, using the distance functions, ERP and ED, as similarity functions.

2. SimTraj

SimTraj is a trajectories storage method in metric spaces, which has a distance-based indexing. This means that the trajectories are grouped into partitions, based on distances, between a set of selected trajectories and the remaining trajectories. At search time, the space partitions enable the discarding/retaining of some subsets, without additional calculations of distance, based on the metric properties of the similarity function. This method is a combination of two data structures, RLC (*distance-based partition*), which organizes the trajectories in clusters based on distance, and an in-memory structure (*frontline*), which stores the pointers to the clusters where the trajectories are stored, as showed in Figure 1.

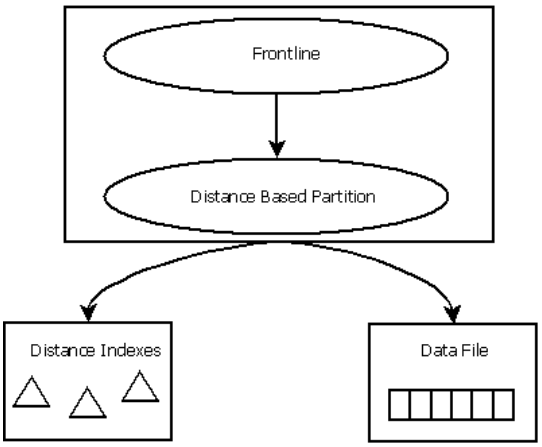

Figure 1. SimTraj Diagram

The *frontline* is used to provide efficient updates in the SimTraj. The *distance-based partition* is used to provide efficient similarity queries, and is implemented with recursive lists of clusters (RLC). A RLC cluster is a triple <c, r, I>, where c is the centre, r the radius and I the interior of cluster. The interior is composed by elements whose distance to the centre is a value equal or less than the radius, and may be implemented as a list of clusters or as a leaf, depending on the number of elements and on the RLC capacity. Figure 2 shows how the trajectories are stored in the SimTraj method. In this figure, it is possible to see in (a) the partitions of trajectories in the space, and in (b) the organization of these trajectories in SimTraj, which has a RLC with capacity five and with variable radius at clusters.

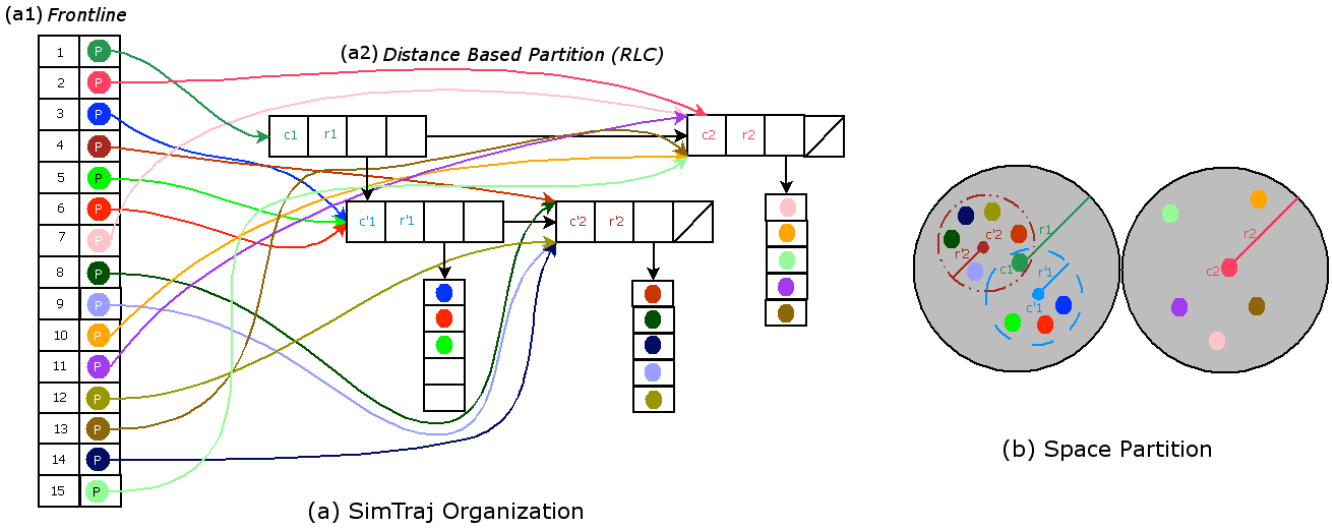

Figure 2. Organization of trajectories in SimTraj

In SimTraj, the insertion of a new trajectory consists in searching for a RLC cluster, in which the distance between the new trajectory and its center is less than or equal to its radius. If it is not found, a new cluster is created. Otherwise, one of two situations can happen: (1) the cluster is a leaf; (2) the cluster is a list of clusters. In (2) the process is applied recursively until it reaches (1). In (1), the first step is to verify if the leaf is not full (RLC capacity). If this is the case, the new trajectory is stored in the leaf. Otherwise, the leaf is transformed into a clusters list and the trajectory is allocated in one of the clusters. Finally, in both cases, the pair composed by the trajectory and the pointer to the RLC cluster is added to *frontline*.

The kNN query is based on a range query (RQ). To perform a kNN search, k elements are obtained from the RLC iterator and sorted downwardly by the distance to the query trajectory. Then a RQ is realized based on the query trajectory and on the largest distance found, which will be used as the query radius. The range query consists in iterating all RLC clusters, and, for each cluster, in finding the trajectories that lie at a distance from the query trajectory that is lower than or equal to a given value (query radius). Using the properties (triangle inequality and symmetry) of the metric function, many of these groupings of trajectories can be immediately discarded or retained, without additional calculations.

To remove a trajectory, *frontline* is consulted in order to obtain the pointer to the RLC cluster that contains the trajectory. When removing from the RLC, one of two situations can happen: (1) the trajectory is the center of a cluster; (2) the trajectory is stored in a leaf. In (1), the cluster is removed from the list, and all trajectories in this cluster will be inserted at the list of remaining clusters. In (2), the trajectory is removed from the leaf. In both cases, the pair associated to the removed trajectory is deleted from *frontline*.

In SimTraj, an update of a given trajectory (a new position in xy-plane at a given time) consists in removing the trajectory followed by the insertion of the updated trajectory.

It should be noted that insertions and deletions of trajectories that lead to a change in the organization of the remaining trajectories (e.g., the removal of a cluster and the reinsertion of its

elements, or the conversion of a leaf in a list of clusters), require an update in *frontline*, for each trajectory, which changed its location in RLC.

A more elaborated description of the RLC can be consulted at (Mamede 2005).

3. Experiments

In this section, we present an experimental evaluation realized in SimTraj, in order to assess the efficiency of k-NN searches (with k=1 and k=5). This evaluation involves two mechanisms in the *distance-based partition* of SimTraj. These techniques are the use of two metric data structures: RLC and M-tree.

The metric spaces, used in the evaluation, were defined on a hurricanes dataset, which contains all the Atlantic tropical hurricanes between 1851 and 2009 (Unisys 2010) (Figure 3 illustrates 2005).

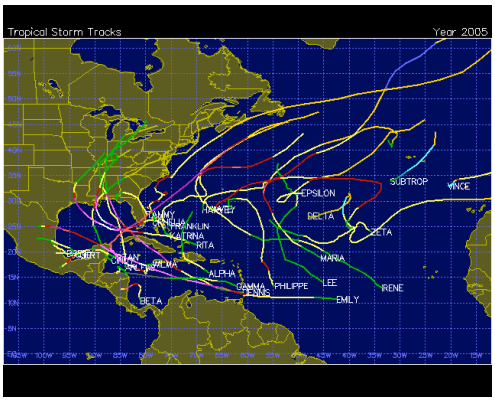

Figure 3. Atlantic tropical hurricanes (year 2005)

In our experiment, the measure of the similarity between two trajectories is based on two distance functions (ERP - Edit distance with Real Penalty, and ED - Euclidean distance). These functions give real values, which represent the degree of similarity between the trajectories. The smaller the function result is, the greater the similarity between the trajectories. ED is L_p-norm with p = 2, and ERP can be viewed as a variant of L_1-norm, which can support local time shifting. To cope with local time shifting, ERP uses an idea from the string edit distance, which represents the number of insert, delete, or replace operations needed to change a string into another string. Note that, in the string edit distance, an added symbol is referred to as a gap element. ERP uses real penalty between two non-gap elements, but a constant value for computing the distance for gaps (origin in the x-y plane). These two functions are metric, as demonstrated in (Chen 2005).

Let $S = <(t_{1s},x_{1s},y_{1s}), \ldots, (t_{ns},x_{ns},y_{ns})>$ and $T = <(t_{1t},x_{1t},y_{1t}), \ldots, (t_{mt},x_{mt},y_{mt})>$ be two trajectories, let $p = (x_p,y_p)$ and $q = (x_q,y_q)$ points in x-y plane and let d be the distance between two points p and q in the x-y plane, denoted by d(p,q).

The Euclidean distance between S and T, denoted by ED(S,T), is defined in Equation 1. This function can only by applied to trajectories that have the same length (n=m). As the length of both trajectories has to be the same, the smaller trajectory needs to be extended. This extension is performed by inserting the start/end point of the small trajectory (point in x-y plane) at the start/end of the sequence, using the respective times of the trajectory with the largest length.

$$ED(S,T) = \sqrt{\Sigma_{1..n}((x_{it}-x_{is})^2 + (y_{it}-y_{is})^2)} \quad (1)$$

The Edit distance with Real Penalty between S and T, denoted by ERP(S,T), is defined in Equation 2.

$$ERP(S,T) = \Sigma_{1..m}\, d((x_{it},y_{it}),(0,0)),\ \text{if } n = 0;$$

$$ERP(S,T) = \Sigma_{1..n}\, d((x_{is},y_{is}),(0,0)),\ \text{if } m = 0;$$

$$\begin{aligned} ERP(S,T) = \min\ \{&ERP(<(t_{2s},x_{2s},y_{2s}), \ldots, (t_{ns},x_{ns},y_{ns})>, <(t_{2t},x_{2t},y_{2t}), \ldots, (t_{mt},x_{mt},y_{mt})>) \\ &+ d((x_{1s},y_{1s}), (x_{1t},y_{1t})), ERP(<(t_{1s},x_{1s},y_{1s}), \ldots, (t_{ns},x_{ns},y_{ns})>, <(t_{2t},x_{2t},y_{2t}), \ldots, (t_{mt},x_{mt},y_{mt})>) \\ &+ d((0,0), (x_{1t},y_{1t})), ERP(<(t_{2s},x_{2s},y_{2s}), \ldots, (t_{ns},x_{ns},y_{ns})>, <(t_{1t},x_{1t},y_{1t}), \ldots, (t_{mt},x_{mt},y_{mt})>) \\ &+ d((x_{1s},y_{st}),(0,0))\}, \text{otherwise} \end{aligned} \quad (2)$$

RLC and M-Tree were parameterized in order to have the most efficient performance in the k-NN searches. While the values of RLC parameters come from experimental tests, the values of the M-Tree come from the results obtained in (Ciaccia et al. 1997).

In our experiment, we perform 37 (25% of the size database) kNN, for each k (k =1 and k=5). The trajectories chosen to perform the searches were chosen randomly from the dataset.

For each search, we calculated the number of disk accesses (SR), the execution time (ET) and the number of distance calculations performed (SD). Figures 4 and 5 show the average results obtained for each search in 1NN and 5NN, respectively.

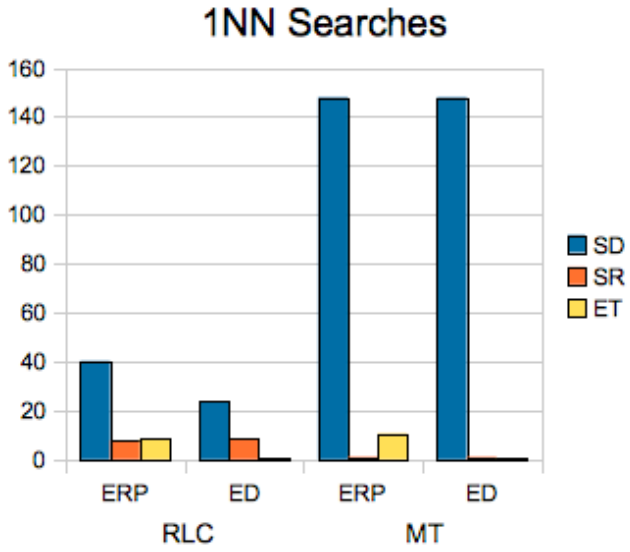

Figure 4. 1NN Searches using ERP and ED at both Data Structures

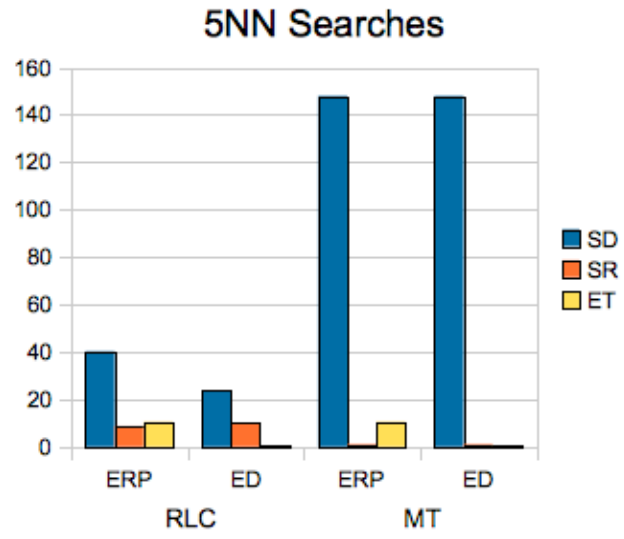

Figure 5. 5NN Searches using ERP and ED at both Data Structures

We can conclude that the RLC and the M-Tree are very competitive. However, the RLC is better at the number of distance calculations and at the execution time. The M-Tree has better values in disk accesses.

Based on disk accesses, one could imagine that the M-Tree would be unbeatable. However such was not true. This happens due the fact that the M-Tree only accesses the disk once, but

performs a higher number of distances calculations than the RLC. Similar results with a different trajectories dataset were obtained in (Afonso et al. 2011).

5. Conclusions and Future Work

In this work we present a trajectories storage method (SimTraj) for efficient similar search in metric spaces. The choice of the RLC data structure to distance-based partition from SimTraj was based on experimental tests performed on two data sets, hurricanes and buses.

An ongoing work is the evaluation the dynamism of the SimTraj. As future work, we aim to explore some interesting scenarios related to the comparison of metric data structures with non-metric ones, as well as the evaluation of SimTraj in a concrete application.

6. References

Afonso, F., Barbosa, F. & Rodrigues, A., 2011. Trajectory Data Similarity with Metric Data Structures. In *Proceedings of GISRUK 2011*. Portsmouth, United Kingdom.

Chávez, E. et al., 2001. Searching in Metric Spaces. *ACM Computing Surveys (CSUR)*, 33(3).

Chen, L., 2005. *Similarity-based Search Over Time Series and Trajectory Data*. Ph.D. Thesis. Waterloo, Ontario, Canada: University of Waterloo.

Ciaccia, P., Patella, M. & Zezula, P., 1997. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB 1997)*. Athens, Greece: Morgan Kaufmann Publishers, pp. 426-435.

GBDI, 2009. M-Tree. *Databases and Images Group*. Available at: http://www.gbdi.icmc.usp.br/en/home.

Mamede, M., 2005. Recursive Lists of Clusters: a Dynamic Data Structure for Range Queries in Metric Spaces. In *Proceedings of the 22th International Symposium on Computer and Information Sciences*. Istambul, Turkey: Springer-Verlag, pp. 843-853.

Samet, H., 2006. Foundations of Multidimensional and Metric Data Structures. In *Foundations of Multidimensional and Metric Data Structures*. USA: Morgan Kaufmann Publishers Inc., pp. 50-89; 270-311; 356-357; 566; 608.

Sarmento, Â.M.L., 2010. *Estruturas de Dados Métricas Genéricas Memória Secundária*. Master's Thesis in Computer Engineering. Caparica, Portugal: UNL/FCT (in Portuguese).

Unisys, 2010. Hurricanes Dataset. *Atlantic Tropical Storm Tracking by Year*. Available at: http://weather.unisys.com/hurricane/atlantic/.