Journal of Systems and Software (accepted), 2008

Searching for Similar Trajectories in Spatial Networks

E. Tiakas A.N. Papadopoulos A. Nanopoulos Y. Manolopoulos
Department of Informatics, Aristotle University
54124 Thessaloniki, GREECE
{tiakas,apostol,alex,manolopo}@delab.csd.auth.gr

Dragan Stojanovic Slobodanka Djordjevic-Kajan
Department of Computer Science, University of Nis
Aleksandra Medvedeva 14, 18000 Nis, SERBIA and MONTENEGRO
{dragans,sdjordjevic}@elfak.ni.ac.yu

Abstract

In several applications, data objects move on predefined spatial networks such as road segments, railways, and invisible air routes. Many of these objects exhibit similarity with respect to their traversed paths, and therefore two objects can be correlated based on their motion similarity. Useful information can be retrieved from these correlations and this knowledge can be used to define similarity classes. In this paper, we study similarity search for moving object trajectories in spatial networks. The problem poses some important challenges, since it is quite different from the case where objects are allowed to move freely in any direction without motion restrictions. New similarity measures should be employed to express similarity between two trajectories that do not necessarily share any common sub-path. We define new similarity measures based on spatial and temporal characteristics of trajectories, such that the notion of similarity in space and time is well expressed, and moreover they satisfy the metric properties. In addition, we demonstrate that similarity range queries in trajectories are efficiently supported by utilizing metric-based access methods, such as M-trees.

Keywords: *spatial networks, moving objects, trajectories, similarity search*

1 Introduction

In location-based services it is important to query the underlying objects based on their location in space, which may change with respect to time. To support such services from the database point of view, specialized tools are required which enable the effective and efficient processing of queries. Queries may involve the spatial or temporal characteristics of the objects, or both (spatio-temporal queries) [26, 22]. Evidently, indexing schemes are ubiquitous to efficiently support queries on moving objects, by quickly discarding non-relevant parts of the database.

We distinguish between two different research directions towards query processing in moving objects, which differ both in the type of queries supported and the characteristics of the indexing schemes used in each case:

[I] Query processing techniques for past positions of objects, where past positions of moving objects are archived and queried, using multi-version access methods or specialized access methods for object trajectories [13, 15, 17, 20, 21]. By studying the past positions of objects, important conclusions can be obtained regarding their mobility characteristics.

The difficulty in this case is that the database volume increases considerably, since new positions are tracked and recorded.

[II] Query processing techniques for present and future positions of objects, where each moving object is represented as a function of time, giving the ability to determine its future positions according to the current motion characteristics of objects (reference position, velocity vector) [8, 9, 27, 18, 10]. These methods are mainly used to support queries according to the current positions and enable predictions of their future locations. The difficulty in this case is to perform effective predictions, which is difficult taking into consideration that some positions will be invalidated, due to changes in the speed and direction of some objects in the near future.

A data set of moving objects is composed of objects whose positions change with respect to time (e.g., moving vehicles). Since in many cases only the position of each object is important, moving objects are modeled as *moving points* in 2-D or 3-D Euclidean space. Queries that involve a particular time instance are characterized as *time-slice queries*, whereas queries that must be evaluated for an interval $[t_s, t_e]$ are characterized as *time-interval queries.* The research community has studied both types extensively. Examples of basic queries that could be posed to such a data set include:

- *Window query*: given a rectangle R, which may change position and size with respect to time, determine the objects that are covered by R from time point t_s to t_e.
- *Nearest-neighbor query*: given a moving point P determine the k nearest-neighbors of P within the time interval $[t_s, t_e]$.
- *Join query*: given two moving data sets U and V, determine the pairs of objects (o_1, o_2) with $o_1 \in U$ and $o_2 \in V$ such that o_1 and o_2 overlap at some point in $[t_s, t_e]$.

Apart from the query processing techniques proposed for the fundamental types of queries (i.e., window, k-NN and join), the issue of *trajectory similarity* has been studied recently. The problem is to identify similar trajectories with respect to a given query trajectory.

The common characteristic of the aforementioned approaches and research works is that objects are allowed to move freely in 2-D or 3-D space, without any motion restrictions. However, in a large number of applications, objects are allowed to move only on predefined paths of an underlying network, resulting in constraint motion. For example, vehicles in a city can only move on road segments. In such a case, the Euclidean distance between two moving objects does not reflect their real distance. Figure 1 shows such an example which illustrates the differences between restricted and unrestricted trajectories. Objects moving in a spatial network follow specific paths determined by the graph topology, and therefore arbitrary motion is prohibited. This means that two trajectories which are similar regarding the Euclidean distance may be dissimilar when the network distance is considered. The majority of existing methods for trajectory similarity assume that objects can move anywhere in the underlying space, and therefore do not support motion constraints. Most of the proposals are inspired by the time series case, and provide translation invariance, which is not always meaningful in the case of spatial networks. To attack this problem, the network is modeled as a directed graph, and the distance between two objects is evaluated by using algorithms for shortest paths between the nodes of the graph.

Therefore, the challenge is to express trajectory similarity by respecting network constraints, which is also a strong motivation for the following real and practical applications:

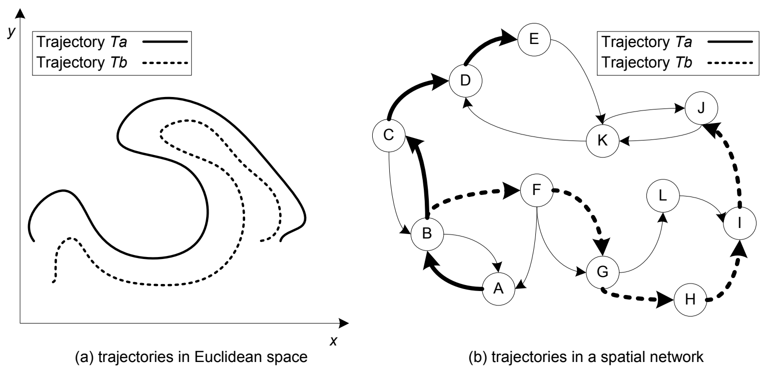

y
x
Trajectory Ta
Trajectory Tb
(a) trajectories in Euclidean space
A
B
C
D
E
F
G
H
I
J
K
L
Trajectory Ta
Trajectory Tb
(b) trajectories in a spatial network

Figure 1: Trajectories in (a) 2-D Euclidean space, and (b) in a spatial network.

[I] By identifying similar trajectories, effective data mining techniques (e.g., clustering) can be applied to discover useful patterns. For example, a dense cluster is an indication of emerge traffic measures, future road expansions, traffic-jam detection, traffic predictions, etc.

[II] Trajectory similarity can also help in several road network applications such as, routing applications which support historical trajectories, logistic applications, city emergency handling, drive guiding systems, flow analysis etc. In such applications, efficient indexing and query processing techniques are required.

[III] Trajectory similarity of moving objects resembles path similarity of user *click-streams* in the area of *web usage mining*. By analyzing the URL path of each user, we are able to determine paths that are very similar, and therefore effective caching strategies can be applied. In web usage mining, web pages and URL links are modeled as a graph. A node in the graph represents a web page, and an edge from one page to another represents an existing link between them. The time spent by each user to a page is also recorded, and it is used in expressing path similarity, in addition to the number of common web pages along each path. In the existing approaches, two paths are considered similar only if they share at least one common web page, or if the paths contain web pages with similar concept. In trajectory similarity on the other hand, two trajectories may be characterized similar even if they do not share any nodes. Therefore, the existing web usage mining techniques are not directly applicable, and the detection of network trajectory similarities can accelerate the web usage mining queries.

The rest of the article is organized as follows. In the next section, we give the appropriate background, we present related work. In Section 3, trajectory similarity search is presented by investigating effective similarity measures between trajectories in a spatial network. Indexing and query processing issues are covered in Section 4, whereas Section 5 offers experimental results. Finally, Section 6 concludes the work.

2 Related Work

In several applications, the mobility of objects is constrained by an underlying spatial network. This means that objects cannot move freely, and their position must satisfy the network constraints. Network connectivity is usually modeled by using a graph representation, composed of a set of vertices (nodes) and a set of edges (connections). Depending on the application, the graph may be *weighted* (a cost is assigned to each edge) and *directed* (each edge has an orientation). Figure 2 illustrates an example of a spatial network corresponding to a part of a city road network, and its graph representation.

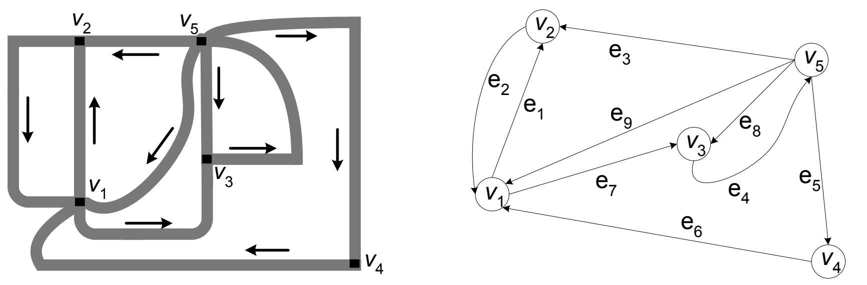


Figure 2: A road network and its graph representation.

Several research efforts have been performed towards efficient spatial and spatio-temporal query processing in spatial networks. In [19] nearest-neighbor query processing is achieved by using a mapping technique. This mapping transforms the graph representation of the network to a high-dimensional space, where Minkowski metrics can be used. Nearest-neighbor queries in road networks have been also studied in [7], where a graph representation is used to model the network. In [16] authors study query processing for stationary data sets, by using both a graph representation for the network and a spatial access method. It is shown that the use of Euclidean distance retrieves many candidates, and instead they propose a network expansion method to process range, nearest-neighbor and join queries. In-route nearest-neighbor queries have been studied in [29], where given a trajectory source and destination the smallest detour is calculated.

The above contributions deal with efficient spatial or spatio-temporal query processing of fundamental queries like range, nearest-neighbor and join. However, the issue of trajectory similarity has not yet been studied adequately in the case of moving objects in spatial networks. Let T_a and T_b be the trajectories of moving objects o_a and o_b respectively, and $D(T_a, T_b)$ a function that expresses their dissimilarity in the range [0, 1]. If the two objects have similar trajectories we expect the value $D(T_a, T_b)$ to be close to zero. On the other hand, if the two trajectories are completely dissimilar, we expect the value $D(T_a, T_b)$ to be close to one.

An example is illustrated in Figure 3, where four trajectories are depicted in the 2-D Euclidean space. A circle denotes the position of each moving object at the corresponding time instance (t_1,...,t_8). It is evident that one expects that the two gray-colored trajectories be very similar, in contrast to the two black-colored trajectories.

In several research proposals, trajectory similarity is viewed as the multidimensional counterpart of time series similarity. In [12] the authors study the problem of similarity search in multidimensional data sequences, to determine similarities in image and video databases. A similarity model based on the Minkowski distance is defined, and each sequence is partitioned to subsequences by means of MBRs, to enable efficient indexing. This work can be viewed as an extension of the method proposed in [3] for time series data.

In [28] a similarity distance between trajectories is defined, which is invariant to translation, rotation and scaling. Again, the distance calculation is based on the Minkowski distance. Objects are allowed to move freely in the address space.

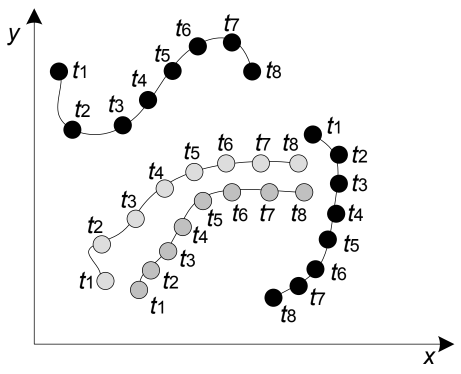


Figure 3: Example of four trajectories in the 2-D Euclidean space.

In [14] an approach is studied to aggregate similar trajectories using a grid-based spatial unit aggregation. The notion of spatial similarity lies on the neighboring cells of the grid in a standard 2-dimensional Euclidean space. Many problems can be arisen with how the grid must be defined, what the cell dimensions must be, and in objects and clusters identification.

In [11] an efficient algorithm for trajectories similarity calculation is presented. But all distance calculations through trajectories are based on Euclidean metrics and spaces (L_p norms).

The method proposed in [24, 25] employs a similarity distance based on the longest common subsequence (LCS) between two trajectories. This approach proposes a distance measure, which is more immune to noise than the Minkowski distance, but does not satisfy the metric space properties, and therefore it is difficult to exploit efficient indexing schemes. Instead, hierarchical clustering is used to group trajectories. Moreover, the similarity measure depends heavily on two parameters, namely δ and ϵ, which must be known in advance, and cannot be altered dynamically without reorganization. These values determine the maximum distance between two locations of different trajectories, in time and space respectively, to be characterized as similar. Trajectories that differ more are characterized as dissimilar and therefore their similarity is set to zero. This approach does not permit the use of ranking or incremental computation of similarity nearest-neighbor queries.

To the best of the authors' knowledge, the only research work studying trajectory similarity on networks is the work in [5, 6]. The authors propose a simple similarity measure based on POIs (points of interest). They retrieve similar trajectories on road network spaces and not in Euclidean spaces. They propose a filtering method based on spatial similarity and refining similar trajectories based on temporal distance. In order to determine the spatial similarity between trajectories, they define that two trajectories are similar in space by a set of pre-defined points of interest P if all points of P lie in both trajectories, otherwise they define the two trajectories as dissimilar. There are several drawbacks using this approach:

- The set of points of interest must be pre-defined and controlled by the user which is very restrictive.
- A simple wrong point selection in P can harm trajectory spatial similarity and the derived similarity clusters, so points in P must be selected very carefully and by an expert of the used road network.
- The similarity in space with such definition (1=similar, 0=dissimilar) does not take into account any notion of similarity percentage or similarity range. Therefore, we cannot determine how similar two trajectories are in space.

- The spatial similarity of two trajectories is based only into the fact that they share common points, and not into the general network space. Therefore, many similarities excluded. For example, trajectories that have parallel edges with only a city block distance and no common points, are considered completely dissimilar.

In addition, no details are given with respect to the access methods required to provide efficient similarity search. Moreover, no discussion is performed regarding the metric space properties of the proposed distance measures. Our approach avoids all these drawbacks.

In the sequel, we study in detail the proposed similarity model for trajectory similarity search in spatial networks aiming at: (i) the definition of similarity and distance measures between trajectories that satisfy the metric space properties, (ii) the exploitation of the distance between two graph nodes, which is used as a building block for the definition of trajectory similarity, (iii) the incorporation of time information in the similarity metric, and (iv) the efficient support of similarity queries by exploiting appropriate indexing schemes and applying fast processing algorithms.

3 Trajectory Similarity Measures

Let $\mathcal{T}$ be a set of trajectories in a spatial network, which is represented by a graph $G(V, E)$, where V is the set of nodes and E the set of edges. Each trajectory $T \in \mathcal{T}$ is defined as:

$$T = \left((v_1, t_1), (v_2, t_2), ..., (v_m, t_m) \right) \tag{1}$$

where m is the trajectory description length, v_i denotes a node in the graph representation of the spatial network, and t_i is the time instance (expressed in time units, e.g., seconds) that the moving object reached node v_i, and $t_1 < t_i < t_m$, $\forall 1 < i < m$. It is assumed that moving from a node to another comes at a non-zero cost, since at least a small amount of time will be required for the transition. Table 1 gives the most important symbols and the corresponding definitions that are used in our study.

3.1 Expressing Trajectory Similarity

We will follow a step-by-step construction of the similarity measure by first expressing similarity taking into account only the visited path, ignoring time information. Time information will be considered in a subsequent step.

We begin our exploration by assuming that any two trajectories have the same description length. This assumption will be relaxed later. Let T_a and T_b be two trajectories, each of description length m. By using our trajectory definition and ignoring the time information, we have: $T_a = (v_{a1}, v_{a2}, ..., v_{am})$ and $T_b = (v_{b1}, v_{b2}, ..., v_{bm})$, where $\forall i$, $v_{ai} \in V$ and $v_{bi} \in V$.

Note that, to characterize two trajectories as *similar* it is not necessary that they share common nodes. Therefore, the similarity measure must take into account the *proximity* of the trajectories (how close is one trajectory with respect to the other).

Due to motion restrictions posed by the spatial network, measuring trajectory proximity by means of the Euclidean distance is not appropriate. Instead, it is more natural to use the cost associated with each transition from a graph node to another. For example, in Figure 4 we observe that two trajectory parts can be similar regarding the Euclidean distance, but may be dissimilar regarding the shortest path distance (network-distance). Thus, for every pair of points between these two trajectory parts, the Euclidean distance is small, but the corresponding

Symbol	**Description**
$\mathcal{T}$	set of trajectories
$\mathcal{S}$	set of sub-trajectories
T, T_a, T_b	trajectories
T_q	a query trajectory
m	trajectory description length
$G(V, E)$	graph representation of the spatial network
D_G	graph diameter
DE_G	maximum Euclidean node distance
v_i	a node in the graph representation
t_i	time instance that the object reached node v_i
e	an edge of the graph
$T[i].v$	the i-th node of the trajectory
$T[i].t$	the time instance that the object reached the i-th node
$d(v_i, v_j)$	network-based distance between two nodes
$de(v_i, v_j)$	Euclidean distance between two nodes
$D_{netX}(T_a, T_b)$	network-based distance between trajectories
$D_{time}(T_a, T_b)$	time-based distance between trajectories
E_{net}	query radius for network-based similarity
E_{time}	query radius for time-based similarity

Table 1: Basic notations used throughout the study.

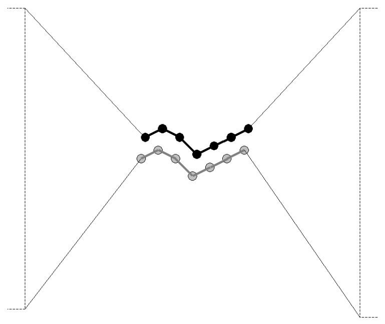


Figure 4: Trajectory proximity.

network-distance is large because the long edges must be crossed. Therefore, it is important in network applications to use the network-distance metric instead of the Euclidean metric.

Let $c(v_i, v_j)$ denote the cost function to travel from a source node v_i to a destination node v_j. As we have already mentioned, this cost for the most network based applications is defined as the shortest path distance (network-distance) between the two nodes. In this paper we fix this cost to be the network-distance. We also fix the following requirements for the graph representation of the network G: *G must be a directed or non-directed, positive weighted and strongly connected graph.* These cases represent successfully the most real network applications (road networks, etc.).

The cost function (network distance) satisfies the following properties:

Property I: *The cost function $c(v_i, v_j)$ gives zero values if and only if $v_i \equiv v_j$.*

It is obvious that $c(v, v) = 0$ for any node v in the graph representation. It also holds that $c(v_i, v_j) = 0 \Rightarrow v_i \equiv v_j$, because it has been assumed that any transition between nodes comes at a non-zero cost (positive weighted graphs).

Property II: *The cost function $c(v_i, v_j)$, definitely satisfies the positivity property and the triangular inequality*:

- $c(v_i, v_j) \geq 0$
- $c(v_i, v_j) \leq c(v_i, v_x) + c(v_x, v_j)$

Property III: *The cost function $c(v_i, v_j)$, does not satisfy in general the symmetric property, therefore it is not definitely a metric function*:

- $c(v_i, v_j) \neq c(v_j, v_i)$

But how does this reflect reality? Consider a directed road network with many one-way road segments, which is quite common. Then, it is clear that if a car goes from a source node v_i to a destination node v_j, it will cover a distance generally different than its way back from v_j to v_i, as it has to pass through different nodes with different weights.

3.1.1 Network Distance Measure 1

The first network distance measure D_{net1} that we propose uses network-based computations. The distance $d(v_i, v_j)$ between any two nodes v_i and v_j, belonging to trajectories T_a and T_b respectively, is given by the following definition.

Definition 1
The distance $d(v_i, v_j)$ between two graph nodes v_i and v_j is defined as follows:

$$d\left(v_i, v_j\right) = \begin{cases} 0 & , if \qquad v_i = v_j \\ \frac{c(v_i,v_j)+c(v_j,v_i)}{2D_G} & , otherwise \end{cases} \tag{2}$$

where $D_G = \max\left\{c\left(v_i, v_j\right), \forall v_i, v_j \in V(G)\right\}$ is the diameter of the graph G of the spatial network and is a global constant for the applications. Its value can be computed taking the overall maximum of possible values of the cost function.

Proposition 1
The distance function $d(v_i, v_j)$ assumes values in the interval [0,1].

Proof
This is obvious when the function returns a zero value. Otherwise it returns the ratio $\frac{c(v_i,v_j)+c(v_j,v_i)}{2D_G}$. But, clearly we have: $c(v_i, v_j) \leq D_G$ and $c(v_j, v_i) \leq D_G$, and by summation we get: $c(v_i, v_j) + c(v_j, v_i) \leq 2D_G$. Therefore, by division we get: $d\left(v_i, v_j\right) = \frac{c(v_i,v_j)+c(v_j,v_i)}{2D_G} \leq 1$. In addition, we have always $c\left(v_i, v_j\right) \geq 0$ and $c\left(v_j, v_i\right) \geq 0$ (positivity), thus $d\left(v_i, v_j\right) \geq 0$. □

Proposition 2

The distance function $d(v_i, v_j)$ satisfies the metric properties.

Proof
We need to prove the following properties for every graph nodes v_i, v_j, v_x:
(i) $d(v_i,\ v_j) \geq 0$
(ii) $d(v_i,\ v_j) = d(v_j,\ v_i)$
(iii) $d(v_i,\ v_j) \leq d(v_i,\ v_x) + d(v_x,\ v_j)$

Clearly, property (i) is true by Proposition 1. Property (ii) is always true if $v_i = v_j$. Otherwise, if $v_i \neq v_j$, we have:

$$d(v_i, v_j) = \frac{c(v_i, v_j) + c(v_j, v_i)}{2D_G} = \frac{c(v_j, v_i) + c(v_i, v_j)}{2D_G} = d(v_j, v_i)$$

Thus, it is true in any case.

Property (iii) is obvious if $v_i = v_j$ or $v_i = v_x$ or $v_j = v_x$. Otherwise, if $v_i \neq v_j \neq v_x$ by substitution we get:

$$\frac{c(v_i, v_j) + c(v_j, v_i)}{2D_G} \leq \frac{c(v_i, v_x) + c(v_x, v_i)}{2D_G} + \frac{c(v_x, v_j) + c(v_j, v_x)}{2D_G} \tag{3}$$

Due to the fact that the cost function satisfies the triangular inequality, we have:

$$c(v_i, v_j) \leq c(v_i, v_x) + c(v_x, v_j)$$

$$c(v_j, v_i) \leq c(v_j, v_x) + c(v_x, v_i)$$

By summation and by division with $2D_G$ we take Inequality (3), thus property (iii) has been proven. □

Definition 2
The network distance $D_{net1}(T_a, T_b)$ between two trajectories T_a and T_b of description length m is defined as follows:

$$D_{net1}\left(T_a, T_b\right) = \frac{1}{m}\sum_{i=1}^{m}\left(d\left(v_{ai}, v_{bi}\right)\right) \tag{4}$$

Proposition 3
The distance measure $D_{net1}(T_a, T_b)$ assumes values in the interval [0,1].

Proof
Omitted □

Proposition 4
The distance measure $D_{net1}(T_a, T_b)$ satisfy the metric properties.

Proof
We need to prove the following properties for every trajectories T_a, T_b, T_x of description length m:
(i) $D_{net1}(T_a,\ T_b) \geq 0$
(ii) $D_{net1}(T_a,\ T_b) = D_{net1}(T_b,\ T_a)$

(iii) $D_{net1}(T_a,\ T_b) \leq\ D_{net1}(T_a,\ T_x) + D_{net1}(T_x,\ T_b)$
Clearly, property (i) is true by consulting Proposition 3. Property (ii) is true because is also true for the distance function d (Proposition 2), so:

$$D_{net}\left(T_a, T_b\right) = \frac{1}{m}\sum_{i=1}^{m}\left(d\left(v_{ai}, v_{bi}\right)\right) = \frac{1}{m}\sum_{i=1}^{m}\left(d\left(v_{bi}, v_{ai}\right)\right) = D_{net}\left(T_b, T_a\right)$$

Property (iii) is written equally by substitution:

$$\frac{1}{m}\sum_{i=1}^{m}\left(d\left(v_{ai}, v_{bi}\right)\right) \leq \frac{1}{m}\sum_{i=1}^{m}\left(d\left(v_{ai}, v_{xi}\right)\right) + \frac{1}{m}\sum_{i=1}^{m}\left(d\left(v_{xi}, v_{bi}\right)\right) \Leftrightarrow$$

$$\Leftrightarrow \sum_{i=1}^{m}\left(d\left(v_{ai}, v_{bi}\right)\right) \leq \sum_{i=1}^{m}\left(d\left(v_{ai}, v_{xi}\right)\right) + \sum_{i=1}^{m}\left(d\left(v_{xi}, v_{bi}\right)\right) \tag{5}$$

From Proposition 2 we have the following inequalities:

$$d\left(v_{ai}, v_{bi}\right) \leq d\left(v_{ai}, v_{xi}\right) + d\left(v_{xi}, v_{bi}\right) \qquad ,\forall i \in \{1, 2, ..., m\}$$

By summation we get (5). □

Figure 5 shows two trajectories T_a, T_b for which we are interested to calculate their distance. Assuming that $D_G = 100$, we have the following calculations:

$$d(v_{ai}, v_{bi}) = \left\{\frac{17}{200}, \frac{16}{200}, \frac{9}{200}, \frac{7}{200}, 0, \frac{5}{200}, \frac{13}{200}\right\}$$

$$D_{net1}\left(T_a, T_b\right) = \frac{1}{7}\left(\frac{17}{200} + \frac{16}{200} + \frac{9}{200} + \frac{7}{200} + 0 + \frac{5}{200} + \frac{13}{200}\right) = \frac{1}{7}\frac{67}{200} = 0.047857$$

3.1.2 Network Distance Measure 2

The second distance measure, D_{net2}, that we propose uses an Euclidean-based distance function (de) in combination with the previous global constant D_G (the graph diameter by the network distance). It can be used for fast calculations only for graphs where the coordinates of the nodes are available. In fact, in many cases the Euclidean distance results in poor performance regarding

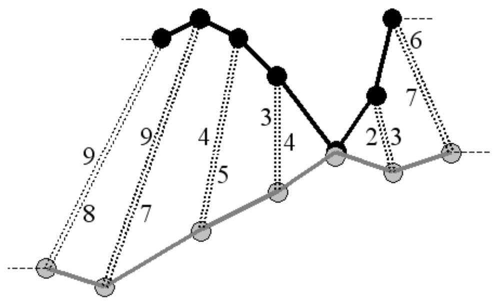


Figure 5: Trajectory similarity example.

the quality of results. However, as it will be described later, it offers a “quick-and-dirty” view of the results.

Definition 6
The distance $de(v_i, v_j)$ between two graph nodes v_i and v_j is defined as follows:

$$de\left(v_i, v_j\right) = \frac{euclidean(v_i, v_j)}{D_G} = \frac{\sqrt{\left(x_{v_i} - x_{v_j}\right)^2 + \left(y_{v_i} - y_{v_j}\right)^2}}{D_G} \tag{6}$$

where x_{v_i}, y_{v_i} are the coordinates of node v_i, and x_{v_j}, y_{v_j} are the coordinates of node v_j.

Proposition 7
The distance function $de(v_i, v_j)$ assumes values in the interval [0,1].

Proof
Let DE_G be the maximum Euclidean distance between all nodes of the graph representing the spatial network: $DE_G = \max\{euclidean(v_i, v_j), \forall v_i, v_j \in V(G)\}$. Then it is obvious that:

$$euclidean(v_i, v_j) \leq DE_G \leq D_G, \forall v_i, v_j \in V(G)$$

The last inequality holds because all network distances are always greater than or equal to the corresponding Euclidean distances. Therefore, we have:

$$\frac{euclidean(v_i, v_j)}{D_G} \leq 1 \Leftrightarrow de(v_i, v_j) \leq 1$$

Moreover, as all distances are positive (or zero when $v_i = v_j$), we have always: $de(v_i, v_j) \geq 0$. □

Proposition 8
The distance function $de(v_i, v_j)$ satisfies the metric properties.

Proof
Due to the fact that the Euclidean distance $euclidean(v_i, v_j)$ satisfies the metric properties and $de(v_i, v_j)$ is the Euclidean distance divided by the positive constant D_G, it is evident that $de(v_i, v_j)$ also satisfies the metric properties. □

Definition 7
The network distance $D_{net2}(T_a, T_b)$ between two trajectories T_a and T_b of description length m is defined as follows:

$$D_{net2}\left(T_a, T_b\right) = \frac{1}{m} \sum_{i=1}^{m} \left(de\left(v_{ai}, v_{bi}\right)\right) \tag{7}$$

Proposition 9
The distance measure $D_{net2}(T_a, T_b)$ assumes values in the interval [0,1].

Proof
Omitted □

Proposition 10

The distance measure $D_{net2}(T_a, T_b)$ satisfy the metric properties.

Proof
Omitted $\square$

3.2 Incorporating Time Information

The similarity measures defined in the previous section take into consideration only the traveling cost information, which depends on the spatial network. In applications such as traffic analysis, the time information associated with each trajectory is also very important.

Definition 8
Given two trajectories $T_a \in T$ and $T_b \in T$ of description length m, their distance with respect to time $D_{time}(T_a, T_b)$ is given by:

$$D_{time}(T_a, T_b) = \frac{1}{m-1} \sum_{i=1}^{m-1} \frac{\left|(T_a[i+1].t - T_a[i].t) - (T_b[i+1].t - T_b[i].t)\right|}{\max\left\{(T_a[i+1].t - T_a[i].t), (T_b[i+1].t - T_b[i].t)\right\}}$$

Essentially, the time similarity between two trajectories, as it has been defined, measures their resemblance with respect to the time required to travel from one node to the next (inter-arrival times).

Figure 6 depicts some examples for the time similarity calculations, where we have three trajectory parts T_a, T_b, T_c with the same description length and the inter-arrival times appear next to their directed edges.

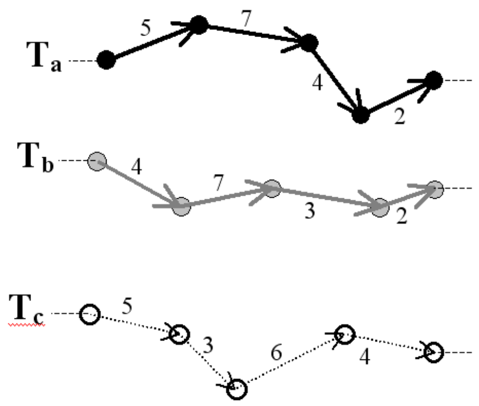


Figure 6: Time similarity calculation example.

With the previous definition we have the following calculations:

$$D_{time}(T_a, T_b) = \frac{1}{4}\left(\frac{1}{5} + \frac{0}{7} + \frac{1}{4} + \frac{0}{2}\right) = 0.1125$$

$$D_{time}\left(T_a, T_c\right) = \frac{1}{4}\left(\frac{0}{5} + \frac{4}{7} + \frac{2}{6} + \frac{2}{4}\right) = 0.35119$$

We observe that T_a is more similar to T_b than T_c and this happens because the corresponding inter-arrival times of the pair T_a, T_b are much closer.

Proposition 11
The distance measure $D_{time}(T_a, T_b)$ assumes values in the interval [0,1].

Proof
Omitted □

Proposition 12
The distance measure $D_{time}(T_a, T_b)$ satisfy the metric properties.

Proof
We need to prove the following properties for any trajectories T_a, T_b, T_x of description length m:

(i) $D_{time}(T_a,\ T_b) \geq 0$

(ii) $D_{time}(T_a,\ T_b) = D_{time}(T_b,\ T_a)$

(iii) $D_{time}(T_a,\ T_b) \leq\ D_{time}(T_a,\ T_x) + D_{time}(T_x,\ T_b)$

Clearly, property (i) is true by Proposition 11. Let us denote the inter-arrival times of all trajectory parts T_a, T_b and T_x as follows: $\delta_{ai} = T_a[i+1].t - T_a[i].t$, $\delta_{bi} = T_b[i+1].t - T_b[i].t$ and $\delta_{xi} = T_x[i+1].t - T_x[i].t$, for all i=1,2,...,m-1. Then, property (ii) is true because we have:

$$D_{time}\left(T_a, T_b\right) = \frac{1}{m-1}\sum_{i=1}^{m-1}\frac{|\delta_{ai} - \delta_{bi}|}{\max\left\{\delta_{ai}, \delta_{bi}\right\}} = \frac{1}{m-1}\sum_{i=1}^{m-1}\frac{|\delta_{bi} - \delta_{ai}|}{\max\left\{\delta_{bi}, \delta_{ai}\right\}} = D_{time}\left(T_b, T_a\right)$$

By substitution, property (iii) is written as:

$$\frac{1}{m-1}\sum_{i=1}^{m-1}\frac{|\delta_{ai} - \delta_{bi}|}{\max\left\{\delta_{ai}, \delta_{bi}\right\}} \leq \frac{1}{m-1}\sum_{i=1}^{m-1}\frac{|\delta_{ai} - \delta_{xi}|}{\max\left\{\delta_{ai}, \delta_{xi}\right\}} + \frac{1}{m-1}\sum_{i=1}^{m-1}\frac{|\delta_{xi} - \delta_{bi}|}{\max\left\{\delta_{xi}, \delta_{bi}\right\}} \Leftrightarrow$$

$$\Leftrightarrow \sum_{i=1}^{m-1}\frac{|\delta_{ai} - \delta_{bi}|}{\max\left\{\delta_{ai}, \delta_{bi}\right\}} \leq \sum_{i=1}^{m-1}\frac{|\delta_{ai} - \delta_{xi}|}{\max\left\{\delta_{ai}, \delta_{xi}\right\}} + \sum_{i=1}^{m-1}\frac{|\delta_{xi} - \delta_{bi}|}{\max\left\{\delta_{xi}, \delta_{bi}\right\}} \tag{8}$$

It is sufficient to prove the following inequalities $\forall\ i = 1, \ldots, m$-1:

$$\frac{|\delta_{ai} - \delta_{bi}|}{\max\left\{\delta_{ai}, \delta_{bi}\right\}} \leq \frac{|\delta_{ai} - \delta_{xi}|}{\max\left\{\delta_{ai}, \delta_{xi}\right\}} + \frac{|\delta_{xi} - \delta_{bi}|}{\max\left\{\delta_{xi}, \delta_{bi}\right\}} \tag{9}$$

To prove 9 it is enough to prove that for every positive numbers a, b, c the following holds:

$$\frac{|a-b|}{\max\left\{a, b\right\}} \leq \frac{|a-c|}{\max\left\{a, c\right\}} + \frac{|c-b|}{\max\left\{c, b\right\}} \tag{10}$$

But, this inequality is obvious if $a = b$, or $a = c$, or $b = c$, and for all other ordering cases of the numbers a, b, c also holds:

- if $a < b < c$ then it gives:

$$\frac{b-a}{b} \leq \frac{c-a}{c} + \frac{c-b}{c} \Leftrightarrow c(b-a) \leq b(c-a) + b(c-b) \Leftrightarrow (b+a)(c-b) \geq 0$$

  which it holds as a, b are positive and $b < c$.

- if $a < c < b$ then it gives:

$$\frac{b-a}{b} \leq \frac{c-a}{c} + \frac{b-c}{b} \Leftrightarrow c(b-a) \leq b(c-a) + c(b-c) \Leftrightarrow (c-a)(b-c) \geq 0$$

  which it holds as $a < c$ and $c < b$.

- if $b < a < c$ then it gives:

$$\frac{a-b}{a} \leq \frac{c-a}{c} + \frac{c-b}{c} \Leftrightarrow c(a-b) \leq a(c-a) + a(c-b) \Leftrightarrow (a+b)(c-a) \geq 0$$

  which it holds as a, b are positive and $a < c$.

- if $b < c < a$ then it gives:

$$\frac{a-b}{a} \leq \frac{a-c}{a} + \frac{c-b}{c} \Leftrightarrow c(a-b) \leq c(a-c) + a(c-b) \Leftrightarrow (c-b)(a-c) \geq 0$$

  which it holds as $b < c$ and $c < a$.

- if $c < a < b$ then it gives:

$$\frac{b-a}{b} \leq \frac{a-c}{a} + \frac{b-c}{b} \Leftrightarrow a(b-a) \leq b(a-c) + a(b-c) \Leftrightarrow (a+b)(a-c) \geq 0$$

  which it holds as a, b are positive and $c < a$.

- if $c < b < a$ then it gives:

$$\frac{a-b}{a} \leq \frac{a-c}{a} + \frac{b-c}{b} \Leftrightarrow b(a-b) \leq b(a-c) + a(b-c) \Leftrightarrow (a+b)(b-c) \geq 0$$

  which it holds as a, b are positive and $c < b$.

Therefore, Inequality (10) is true, and property (iii) has been proven. □

3.2.1 Spatio-temporal Similarity Measures and Methods

We have at hand different distance measures, D_{net} and D_{time}, that can be used to compare trajectories of the same length in space and time. Several applications may require both similarity measures to extract useful knowledge.

There are three different methods in order to retrieve similar trajectories in space-time as proposed in [5]: (i) Searching similar trajectories with direct application of spatio-temporal distance measures, (ii) Filtering trajectories based on temporal similarity and refining similar trajectories based on spatial distance, (iii) Filtering trajectories based on spatial similarity and refining similar trajectories based on temporal distance.

Here we suggest the methods (i) and (iii), due to the fact that method (ii) can hardly be found in practical applications.

To implement method (i) we can combine the two distance measures D_{net} and D_{time} into a single one. For example, the two distances may be weighted with parameters W_{net} and W_{time} such that W_{net}+W_{time}=1. The total (combined) distance can then be expressed as follows:

$$D_{total}\left(T_a, T_b\right) = W_{net} \cdot D_{net}\left(T_a, T_b\right) + W_{time} \cdot D_{time}\left(T_a, T_b\right)$$

It is evident that the distance measure D_{total} satisfies the metric space properties. However, this approach poses a significant limitation, since the values of W_{net} and W_{time} must be known in advance.

Consequently we propose method (iii) using D_{net} and D_{time} separately, where the distance D_{net} making the filtering step in space and the distance D_{time} making the refinement step in time. In this way, two parameter distances are required to be posed by the query. The distance E_{net} expresses the desired similarity with respect to the D_{net} distance measure, whereas the distance E_{time} expresses the desired similarity regarding the D_{time} distance measure. If the user wishes to focus only on the network distance, then the value of E_{time} may be set to 1. Otherwise, another value is required for E_{time}, which determines the desired similarity in the time domain. By allowing the user to control the values of E_{net} and E_{time} a significant degree of flexibility is achieved, since the "weight" of each distance can be controlled at will.

4 Indexing and Query Processing Issues

In this section, we study some important issues regarding trajectory similarity. Firstly, we discuss the problem of handling trajectories of different description length, by decomposing a trajectory to sub-trajectories. Then, we study the use of indexing schemes for sub-trajectories. Finally, we study some fundamental query processing issues.

4.1 Trajectory Decomposition

Up to now we have handled the case where all trajectories are of the same description length. We proceed now to relax this assumption, by considering trajectories of different lengths. In fact, this is the more general case that reflects reality. First of all, two trajectories may involve a different number of visited nodes, and therefore their description length will be different. Furthermore, we cannot always guarantee that moving objects report their positions at fixed time intervals. Due to noise, several measurements may be lost, or different moving objects report their positions at different time intervals. In these cases, two trajectories may have different description lengths.

Let T be a trajectory of description length m. Moreover, let μ denote an integer such that $\mu \leq m$. T is decomposed into m-μ+1 sub-trajectories, by using a window of length μ, and progressively moving one node at a time from left to right. Each of the resulting sub-trajectories has a length of μ. Figure 7 illustrates an example of the decomposition process, where m=6 and μ=3.

By following the same process for all trajectories we get a new set of sub-trajectories S, all of description length μ. Moreover, we have already defined a distance measure for trajectories of the same description length in the previous section given by either D_{net} or D_{time} which both satisfy the metric space properties.

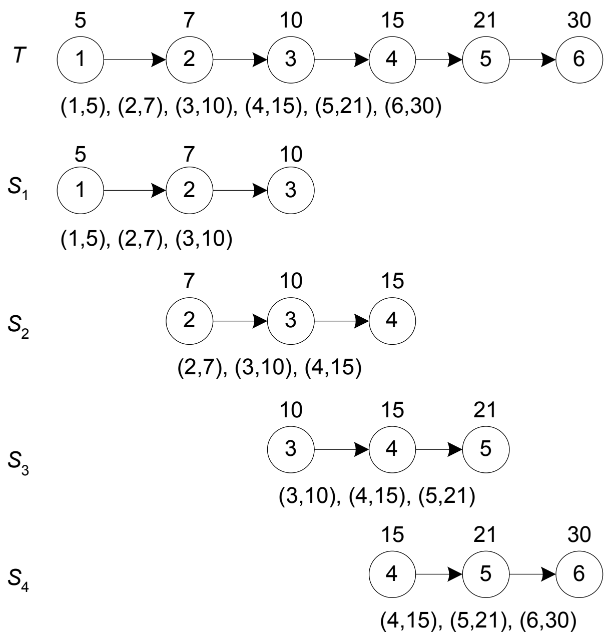


Figure 7: Trajectory decomposition example for $m = 6$ and $\mu = 3$.

4.2 Indexing Schemes

Our next step involves indexing the set $\mathcal{S}$ of sub-trajectories, enabling efficient query processing. Towards this direction, we propose two schemes, which are both based on the M-tree access method [2]. Note that since a vector representation of each sub-trajectory is not available, techniques like R-trees [4] and its variants are not applicable. Recall that, the M-tree is already equipped by the necessary tools to handle range and nearest-neighbor queries, as it has been reported in [2]. The only requirement for the M-tree to work properly is that the distance used must satisfy the metric space properties. Since both D_{net} and D_{time} satisfy these properties, they can be used as distance measures in M-trees. Note that, among the metric indexing schemes we choose the M-tree because of its simplicity. However, other secondary memory schemes for metric spaces or any other metric access method can been applied equally well (e.g., SlimTrees [23]). Two alternatives are followed towards indexing sub-trajectories:

- **M-treeI method**. In this scheme, only the NET-M-tree is used to check the constraint regarding E_{net}. Then, in a subsequent step the candidate sub-trajectories are checked against the time constraints. This way, only one M-tree is used.
- **M-treeII method**. In this scheme, two M-trees are used to handle D_{net} and D_{time} separately. These trees are termed NET-M-tree and TIME-M-tree respectively. Each M-tree is searched separately using E_{net} and E_{time} respectively. Then, the intersection of both results is determined to get the sub-trajectories that satisfy the network and time constraints.

4.3 Query Processing Fundamentals

A user query is defined by a triplet $< T_q, E_{net}, E_{time} >$ where T_q is the query trajectory, E_{net} is the radius for the network distance and E_{time} is the radius for the time distance. For the query processing to be consistent with the proposed framework, each query trajectory T_q must be of at least description length μ. If this is not true, padding is performed by repeating, for example, the last node of the trajectory several times, until the description length μ is reached. In the general case where the description length of T_q is greater than μ, the decomposition process is applied to obtain the sub-trajectories of T_q. Finally, if the description length of T_q is equal to μ, then only one sub-trajectory is produced.

Let p denote the number of sub-trajectories of T_q determined by the trajectory decomposition process. The next step depends on the indexing scheme we utilize, i.e. either M-treeI or M-treeII as they have been described previously. A trajectory is part of the answer if there is at least one of its sub-trajectories that satisfy the network and time constraints for at least one query sub-trajectory. In the sequel, we analyze the whole process in detail:

- Having a query trajectory T_q of description length l and the E_{net}, E_{time} parameters, we decompose T_q into $p = l - \mu + 1$ sub-trajectories (if $l > \mu$) with the window method and then we construct their set $QS(T_q)$.
- For every query sub-trajectory $qs \in QS(T_q)$, we execute a simple range query to NET-M-Tree with radius E_{net} and collect related sub-trajectories into the set C_{net}.
- If M-treeII method is used then we execute another simple range query to TIME-M-Tree with radius E_{time} and collect related sub-trajectories into the set C_{time}.
- If M-treeI method is used then we check every sub-trajectory in C_{net} against E_{time} and from the selected results we construct the set AS. Otherwise, If M-treeII method is used, the results' set AS is constructed with the common sub-trajectories of the sets C_{net} and C_{time}. In both cases, the set AS contains the resulted sub-trajectories ID's.
- From the set AS we take the corresponding trajectories ID's and we construct the final result set AT.

In any case, a trajectory $T \in \mathcal{T}$ will appear in the result set, if and only if there exists at least one sub-trajectory ts of T which is similar to at least one sub-trajectory qs of the query trajectory T_q, and also satisfies the network and time constraints. More formally:

$$T \textit{ is similar to } T_q \Leftrightarrow \exists ts \subseteq T, \exists qs \subseteq T_q : D_{net}(ts, qs) \leq E_{net} \wedge D_{time}(ts, qs) \leq E_{time}$$

Figure 8 presents an outline of the algorithm. Taking into account that the consecutive sub-trajectories of T_q have (μ - 1) common nodes, most calculations and requests can be already in the memory, as we check one sub-trajectory after another, so it is strongly recommended to use an LRU memory buffer.

4.4 Distance Buffering

The distance measure D_{net1} uses the shortest path distance between graph nodes. These computations can be performed more efficiently by using an LRU buffer. The LRU buffer maintains a constant amount of distance values into main memory. In the experimental results section we

```
Algorithm SimilaritySearch($T_q$, $E_{net}$, $E_{time}$, $\mu$)
Input
$T_q$: query trajectory
$E_{net}$: network distance radius
$E_{time}$: time distance radius
$\mu$: minimum description length of query sub-trajectory
Output
$AS$: set of sub-trajectory IDs
$AT$: set of trajectory IDs

1.      $QS(T_q)$ = all sub-trajectories of $T_q$ of description length $\mu$
2.  for each query sub-trajectory $qs \in QS(T_q)$
3.      if method M-treeI is used then
4.          search NET-M-tree using $qs$ and $E_{net}$
5.          $C_{net}$ = candidate sub-trajectories from NET-M-tree
6.          check every sub-trajectory in $C_{net}$ against $E_{time}$
7.          update $AS$
8.      else if method M-treeII is used then
9.          search NET-M-tree using $qs$ and $E_{net}$
10.         $C_{net}$ = candidate sub-trajectories from NET-M-tree
11.         search TIME-M-tree using $qs$ and $E_{time}$
12.         $C_{time}$ = candidate sub-trajectories from TIME-M-tree
13.         $AS = C_{net} \cap C_{time}$
14.     end if
15. end for
16. calculate $AT$ from $AS$
17. return($AS$,$AT$)
```

Figure 8: Outline of similarity search algorithm.

show that only a relatively small buffer size is adequate to accelerate performance, offering a good hit-ratio.

If the network graph has at most a few thousand nodes, it is suggested to precompute all distances $c(v_i, v_j)$ between nodes and to put them into a hash-based file. Then, the LRU memory buffer can cooperate with this file during the request procedure for even better performance. Later, we discuss the alternative of storing only a subset of precomputed distances on the disk, to handle large graphs.

The algorithm in Figure 9 illustrates the process of retrieving a distance $c(v_i, v_j)$. The variables *requests*, *hits*, and *misses* are used to test buffer performance.

It is important to remind that the LRU memory buffer and the precomputed distances disk file, are used only with D_{net1}. They are not necessary for D_{time} calculations and in D_{net2} measure which does not use network distances at all.

4.5 Combining Measures D_{net1} and D_{net2} (Filtering and Refinement)

Due to network restrictions, a similarity range query using the D_{net2} distance measure may return some trajectories that are not similar regarding distance measure D_{net1} (false alarms). This effect is more significant when the shortest path distance between nodes is considerably higher than their Euclidean distance. Therefore, we need to detect these trajectories using another measure, which respects the network restrictions in space, and use it in a refinement step during query processing. For this reason, we can select the distance measure D_{net1} to handle false alarm detection. This procedure will give correct results if and only if we prove that every

```
Algorithm RetrieveDistance($v_i$, $v_j$)
Input
$v_i$: source node
$v_j$: destination node
Output
$c(v_i, v_j)$: value of the cost function between nodes $v_i$ and $v_j$
------------------------------------------------------------
1. requests++
2. search in LRU memory buffer for distance $c(v_i, v_j)$
3. if distance found in buffer then
4.         return($c(v_i, v_j)$)
5.         hits++
6. else
7.         if a precomputed distances disk file is used then
8.              open disk file
9.              find record with distance $c(v_i, v_j)$
10.              return($c(v_i, v_j)$)
11.              insert distance $c(v_i, v_j)$ in memory buffer with LRU rule
12.              misses++
13.         else
14.              compute the distance $c(v_i, v_j)$
15.              return($c(v_i, v_j)$)
16.              insert distance $c(v_i, v_j)$ in memory buffer with LRU rule
17.              misses++
18.         end if
19. end if
```

Figure 9: Outline of distance retrieval algorithm.

trajectory that appears in the result set of D_{net1} measure, appears also in the result set of D_{net2}, when we apply an E_{net} range query.

Proposition 13

For every two trajectories T_a, T_b the following inequality always holds:

$$D_{net2}\left(T_a, T_b\right) \leq D_{net1}\left(T_a, T_b\right)$$

Proof

As the shortest path distance $c(v_i, v_j)$ between two graph nodes v_i, v_j is always greater than or equal to their corresponding Euclidean distance, it always holds that:

$$euclidean(v_i, v_j) \leq c(v_i, v_j), \forall v_i, v_j \in V$$

By dividing with the constant D_G we get:

$$\frac{euclidean\left(v_i, v_j\right)}{D_G} \leq \frac{c(v_i, v_j)}{D_G} \Leftrightarrow de\left(v_i, v_j\right) \leq d\left(v_i, v_j\right), \forall v_i, v_j \in V$$

Therefore, for every two trajectories $T_a = (v_{a1},\ v_{a2},\ ...,\ v_{am})$ and $T_b = (v_{b1},\ v_{b2},\ ...,\ v_{bm})$, where $v_{ai} \in V$ and $v_{bi} \in V$, *(∀i=1,. . . ,m)*, we have the following inequalities:

$$de\left(v_{ai}, v_{bi}\right) \leq d\left(v_{ai}, v_{bi}\right) \qquad , \forall i = 1, ..., m$$

By summation, we get:

$$\sum_{i=1}^{m} \left(de\left(v_{ai}, v_{bi}\right)\right) \leq \sum_{i=1}^{m} \left(d\left(v_{ai}, v_{bi}\right)\right)$$

$$\Leftrightarrow \frac{1}{m}\sum_{i=1}^{m} \left(de\left(v_{ai}, v_{bi}\right)\right) \leq \frac{1}{m}\sum_{i=1}^{m} \left(d\left(v_{ai}, v_{bi}\right)\right)$$

$$\Leftrightarrow D_{net2}\left(T_a, T_b\right) \leq D_{net1}\left(T_a, T_b\right)$$

and the proposition has been proven. □

Following Proposition 13, when we have a query trajectory T_q and a network query range E_{net}, all trajectories returned by D_{net1} measure will appear in the result-set of D_{net2}, because:

$$D_{net2}\left(T_q, T\right) \leq D_{net1}\left(T_q, T\right) \leq E_{net} \quad , \forall T \in \mathcal{T}$$

Figure 10 illustrates the outline of the similarity search algorithm including the refinement step. D_{net2} is used as the filtering distance measure, whereas D_{net1} is used for refinement, to eliminate false alarms. An important observation is that this scheme can be applied to both M-treeI and M-treeII methods, and moreover, it can be used with any well-defined distance measure, as long as the following lower-bounding property holds:

$$D_{filtering}\left(T_a, T_b\right) \leq D_{refinement}\left(T_a, T_b\right) \quad , \forall T_a, T_b \in \mathcal{T}$$

5 Performance Evaluation

In this section, we give information about the implementation of the proposed approach in C++ and the results of experiments that confirm and evaluate all previous algorithms, procedures and techniques. All experiments have been conducted on a Pentium IV running Windows XP, with 1GB of RAM, and a 320GB-SATA2-16MB hard disk. First, we present the construction of used spatial network and trajectory data set. Then, we present the construction of M-Trees for each defined measure and how the proposed measures express well the notion of similarity in space and time. At the main part, we present the evaluation results of all proposed methods for similarity range queries.

5.1 Spatial Network Data

All experiments have been conducted using a real-world spatial network, the road network of Oldenburg city [1]. The cost function $c(v_i, v_j)$ between two nodes of the graph representation is the shortest path distance. The number of vertices in the Oldenburg data set is 6,105. Therefore, the total number of precomputed distances among all possible pairs of vertices is 37,271,025. These distances are stored in a hash-based file on disk (DISTfile), using the Hilbert space filling curve as a hashing function. The Hilbert curve values are derived from the corresponding source and target node ID's of the distances, which are integers into the interval $[0, |V_G| - 1]$, (e.g., for the distance $c(v_i, v_j)$ the value $Hilbert(ID(v_i), ID(v_j))$ is calculated). For the selected road

```
Algorithm SimilaritySearchWithRefinement($T_q$, $E_{net}$, $E_{time}$, $\mu$)
Input
$T_q$: query trajectory
$E_{net}$: network distance radius
$E_{time}$: time distance radius
$\mu$: minimum description length of query sub-trajectory
Output
$ASF$: final set of sub-trajectory IDs
$ATF$: final set of trajectory IDs
------------------------------------------------------------
1.     $QS(T_q)$ = all sub-trajectories of $T_q$ of description length $\mu$
2. $ASF$=Ø
3. for each query sub-trajectory $qs \in QS(T_q)$
4.     if method M-treeI is used
5.        search NET-M-tree (constructed by the basic metric) using $qs$ and $E_{net}$
6.        $C_{net}$ = candidate sub-trajectories from NET-M-tree (using the $D_{net}$ distance of the basic metric)
7.        check every sub-trajectory in $C_{net}$ against $E_{time}$
8.        update $AS$
9.     else if method M-treeII is used then
10.       search NET-M-tree (constructed by the basic metric) using $qs$ and $E_{net}$
11.       $C_{net}$ = candidate sub-trajectories from NET-M-tree (using the $D_{net}$ distance of the basic metric)
12.       search TIME-M-tree using $qs$ and $E_{time}$
13.       $C_{time}$= candidate sub-trajectories from TIME-M-tree
14.       $AS = C_{net} \cap C_{time}$
15.    end if
16.    for each sub-trajectory $S_i$ in $AS$
17.       compute the $D_{net}$ distance of $S_i$ from $qs$ using the selected refinement metric
18.       insert $S_i$ in $ASF$ if that distance is less than or equal to $E_{net}$
19.    end for
20. end for
21. calculate $ATF$ from $ASF$
22. return($ASF$,$ATF$)
```

Figure 10: Outline of similarity search algorithm with refinement step.

network, the total time required for all precomputations and creation of DISTfile is 3,180.581 sec. The record length has been set to 16 bytes, so the final file capacity is 596,336,400 bytes (285MB zipped).

An in-core LRU buffer has been used to keep a number of precomputed distances in main memory (we initialized the buffer selecting some top-used distances through calculations which actually are distances between nodes that included in the most trajectory parts). The size of the buffer has been set to 2,000, which is a relatively small value compared to the total number of pair-wise distances. We have computed the average number of network distance calculation requests, the average number of hits and misses, in simple range queries in space using D_{net1} and D_{net2}. The results show that almost 85% of the distance requests are absorbed by the main memory buffer and therefore, we avoid fetching them from the disk. The more buffer pages are available, the higher the hit ratio becomes.

The fast retrieval of shortest path distances is the most time consuming factor affecting the performance of network-based distance calculations, the construction of M-Trees and finally in the performance of similarity range queries.

5.2 Construction of Trajectories and Sub-trajectories

The trajectory data set T we have used for the experiments consists of 3,797 trajectories of objects moving on the road segments of Oldenburg city, using the generator developed in [1]. Each trajectory has a minimum description length of 10 and maximum description length of 100 nodes. A sliding window of description length $\mu = 10$ has been used to generate the sub-trajectories of each trajectory. Therefore, the total number of sub-trajectories produced (set $\mathcal{S}$) is 75,144.

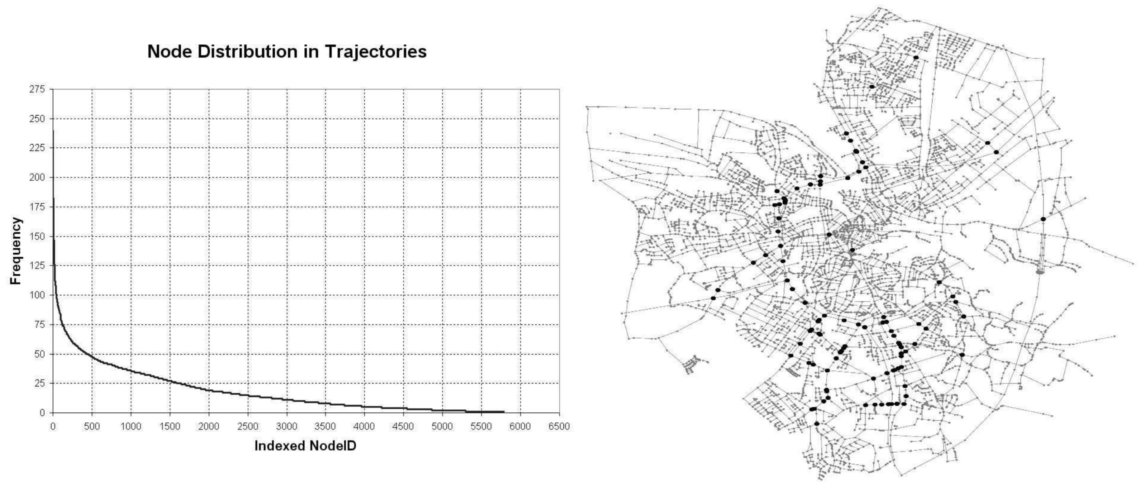


Figure 11: (a) Distribution of nodes in trajectories (b) Top-100 most frequent nodes.

Moreover, it is important to study the distribution of the constructed trajectory data set among the nodes of the road network. This will help to evaluate if the data set represents well a real-world trajectory set of this town. So, we record in a new file all node ID's used by the trajectories, with the frequency that are being used (how many trajectories pass through) in a descending order. Figure 11(a) gives the recorded distribution and Figure 11(b) depicts the top-100 most used nodes in the network by the trajectories.

It is evident that we have a skew distribution of nodes in trajectories and this reflects reality: there are some nodes that are being used very often which are center points of this town or hard traffic points, and the most peripheral nodes are being used much rarely. Therefore, our trajectory data set is a good representative of a real traffic condition.

5.3 M-tree Construction

We have constructed four different M-trees. The NET-M-trees which are implemented based on the $D_{net1,2,3}$ measures and the TIME-M-tree implemented based on the D_{time} measure. Recall that, all M-trees handle the same set S of sub-trajectories of description length μ=10 and not the complete trajectories of moving objects.

We have utilized the bulk-loading method for the construction of all M-trees, and the following parameters values have been used: a page size of 4KB, 5% minimum node utilization, minimum overlaps promote part and root functions, a general hyper-plane split strategy, and radius function by average. Table 2 shows the total number of network distances computed during the construction, the number of zero distances, the final file capacity of M-trees on disk and

M-tree	Distances	Zeros	Capacity	Time
D_{net1}	1,574,890	38,309	32.5MB	13min+7sec
D_{net2}	1,494,416	37,761	32.1MB	35sec
D_{time}	4,013,864	40,461	30.9MB	1min+46sec

Table 2: Information regarding the construction of M-trees.

the total construction time. Note that we have exploited precomputed distances (LRU-buffer and DISTfile) during the construction procedure.

We observe that D_{net2} gives the smallest capacity and construction time, because network distance computations are not required.

5.4 Evaluation of Similarity Measures

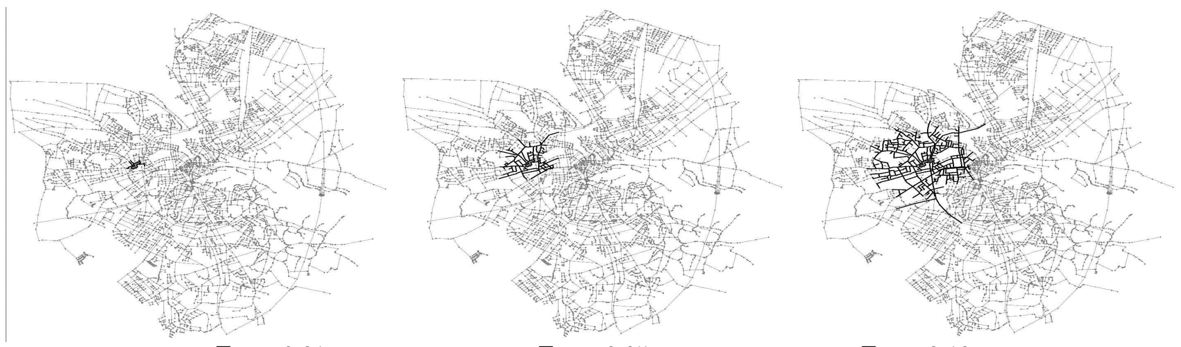
We have randomly selected several trajectories from different areas of Oldenburg and we have performed similarity range queries by using all measures.

Figures 12 and 13 show the results of range queries with radius E_{net} = 0.01, 0.05, 0.10, in a random selected query trajectory from our data set, using the available network distance measures. By studying these figures we observe that:

- In all metrics, the resulted trajectories firstly appeared in the closest neighbor of query trajectory and as the radius E_{net} increases, they expand into connected and almost rounded areas, in which the query trajectory takes a central position.
- All query trajectory results of D_{net1} metric (using a constant E_{net} range) are included in the results of D_{net2} metric, according to Proposition 9.

5.5 Performance Evaluation of M-treeI and M-treeII Methods

In this section, we study the performance of M-treeI,II methods using all the proposed metrics. We selected randomly 100 trajectories from our data set and from different parts of the town and we performed similarity range queries using the M-treeI,II methods. We gave all combination values into the interval [0,1] with a step of 0.05 in E_{net}, E_{time} parameters. The final reported



E_{net}=0.01 E_{net}=0.05 E_{net}=0.10

Figure 12: Preview of range queries using distance measure D_{net1}.

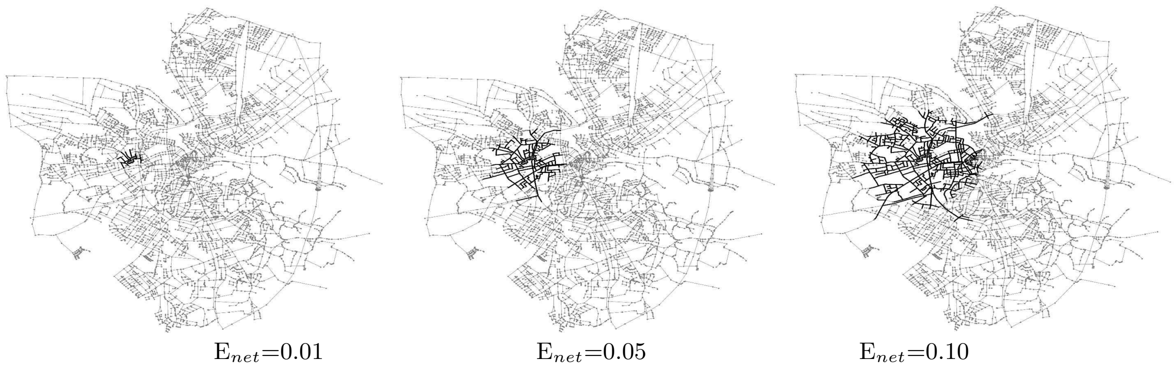


Figure 13: Preview of range queries using distance measure D_{net2}.

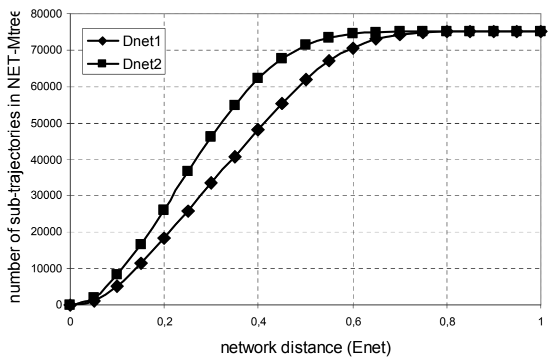
Variable	**Description**
N_{net}	Number of similar sub-trajectories found in NET-M-tree
D_{net}	Number of network-based distance computations
T_{net}	Total searching time in NET-M-tree (sec)
$MBFR$	Memory LRU Buffer Total Requests
$MBFH$	Memory LRU Buffer Total Hits
$MBFM$	Memory LRU Buffer Total Misses
$DBFR$	Disk LRU Buffer Total Requests
$DBFH$	Disk LRU Buffer Total Hits
$DBFM$	Disk LRU Buffer Total Misses
N_{time}	Number of similar sub-trajectories found in TIME-M-tree
D_{time}	Number of time-based distance computations
T_{time}	Total searching time in TIME-M-tree (M-treeII) or in time calculations (M-treeI) (sec)
TT	Total query time
AS	Total number of common (M-treeII) or accepted (M-treeI) sub-trajectories found (Net&Time)
AT	Total number of similar trajectories found (final results)
FA	False alarms for sub-trajectories in D_{net2+1} method

Table 3: Basic variables measured throughout experiments.

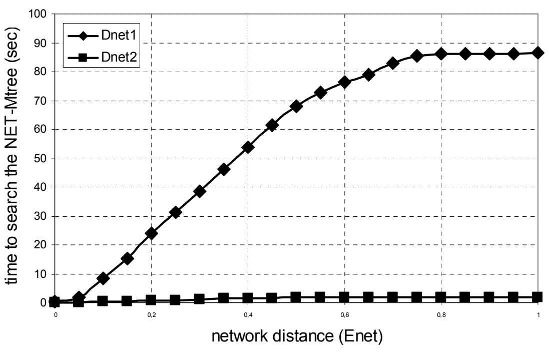
results correspond to the average values of these 100 queries. The basic parameters that are studied are summarized in Table 3.

Figure 14(a) depicts the number of similar sub-trajectories found using all available network-based distance measures. Recall, that the results are the same for both M-treeI and M-treeII methods. As the E_{net} radius increases, D_{net2} first reaches the upper limit (75,144), followed by D_{net1}. Evidently, the distance measure D_{net2} gives more results than D_{net1} due to the lower-bounding property.

Figure 14(b) depicts the total time spent for network-based computations using all network-based distance measures. It is evident that D_{net2} is the less time-consuming measure since distances are computed by using the Euclidean distance of the nodes. The results are similar



(a) number of sub-trajectories



(b) search time

Figure 14: Number of sub-trajectories (a) and search time (b) for NET-M-tree.

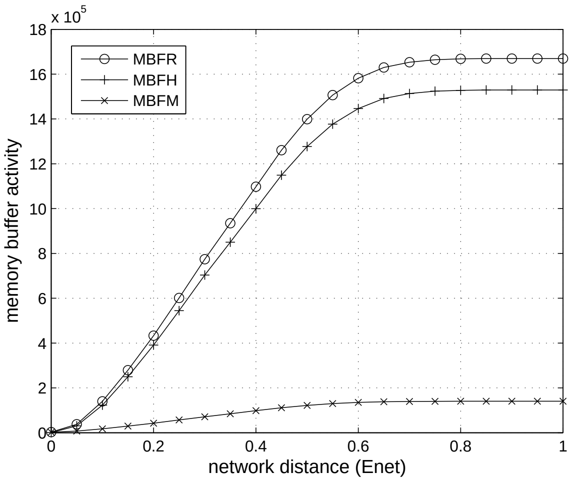


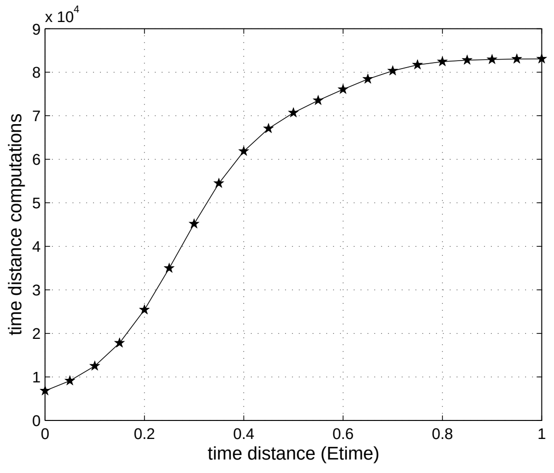
Figure 15: Memory buffer activity for D_{net1}.

for both M-treeI and M-treeII methods.

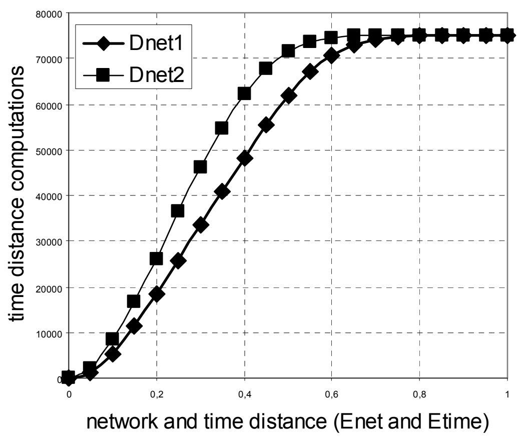
Figure 15 illustrates the memory LRU buffer activity using D_{net1}. Note that D_{net2} does not use the LRU buffer, since no network distance computations are performed. In both cases, the total number of distance hits is about 85%, so with only 2000 buffer pages we have a satisfactory hit ratio. Again, the results are similar for M-treeI and M-treeII methods.

Figure 16 depicts the number of time-based distance computations for M-treeI and M-treeII methods. We observe that in M-treeII method the number of time-based distance computations depends only on the E_{time} radius, whereas in M-treeI method this value depends on both E_{net} and E_{time} parameters, because in this case the total number of computed distances is equal to the number of sub-trajectory results returned by the NET-M-tree. Therefore, we expect less time-based computations in the M-treeI method than in the M-treeII method. This results in slightly better performance for the M-treeI method regarding time-based distance computation overhead, as it is illustrated in Figure 17.

Figure 18 depicts the percentage of false alarms for D_{net2+1}, for various values of parameters E_{net} and E_{time}. In the left part of the figure, these parameters change freely, whereas in the

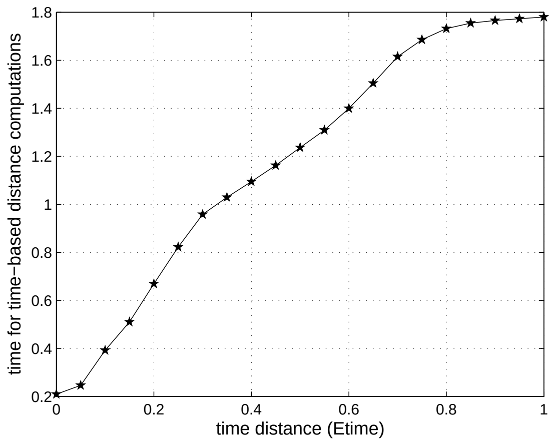


(a) M-treeII

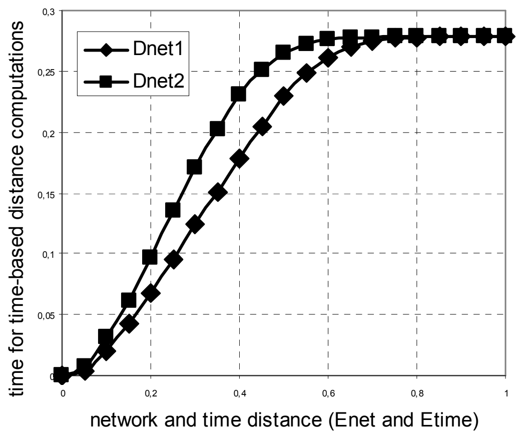


(b) M-treeI

Figure 16: Number of time-based distance computations in M-treeI and M-treeII methods.



(a) M-treeII



(b) M-treeI

Figure 17: CPU time (in sec) required for time-based distance computations in M-treeI and M-treeII methods.

right part always E_{net} equals E_{time}. It is evident, that the existence of false alarms can not be avoided, due to the distance lower-bounding. However, the percentage of false alarms is relatively small, and therefore effective filtering is performed by applying the Euclidean distance prior to network distance computations. The maximum number of false alarms (around 25%) appears when $E_{net} = 0.25$ and $E_{time} = 0.30$.

Figures 19(a) and 19(b) depict some representative results regarding the performance of M-treeI and M-treeII methods for $E_{net} = E_{time}$, using all network distance measures. D_{net2} is the most efficient tool but needs validation of correctness, and D_{net2+1} method is the most attractive alternative that can be used for trajectory similarity search, if efficiency is important. However, care should be taken since the usage of D_{net2+1} involves determination of false alarms. If the number of false alarms is large, performance degradation may appear.

In all the experiments conducted, the method that uses only one M-tree performs marginally

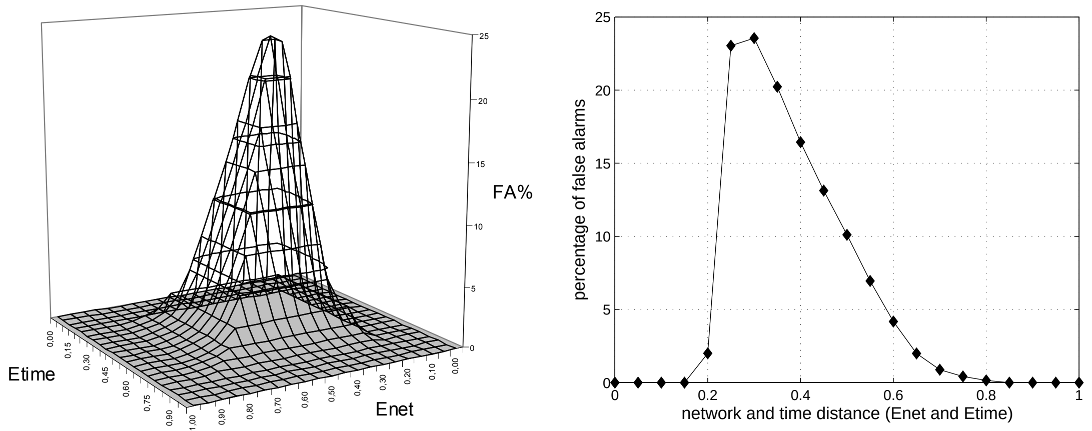


Figure 18: Percentage of false alarms for D_{net2+1} method.

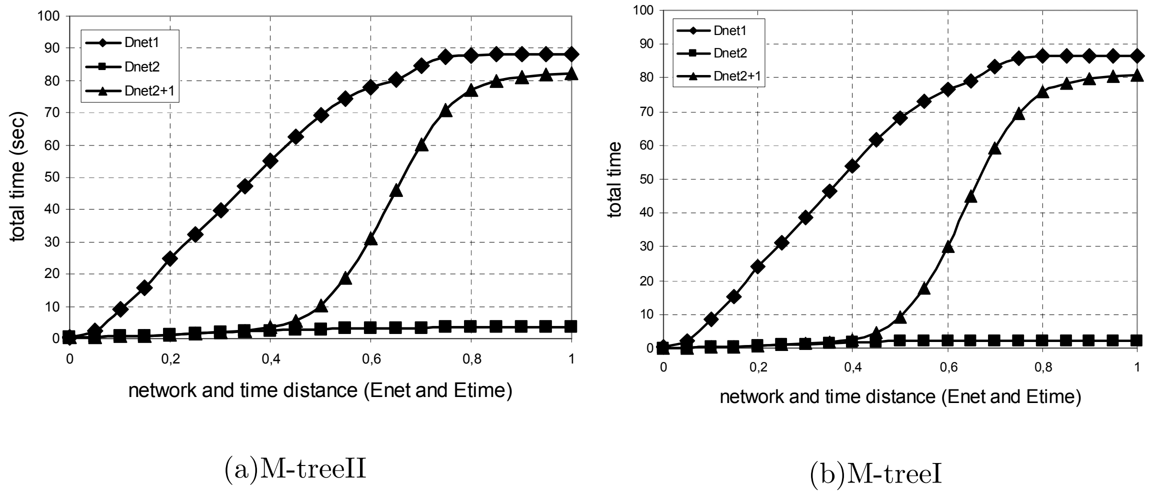
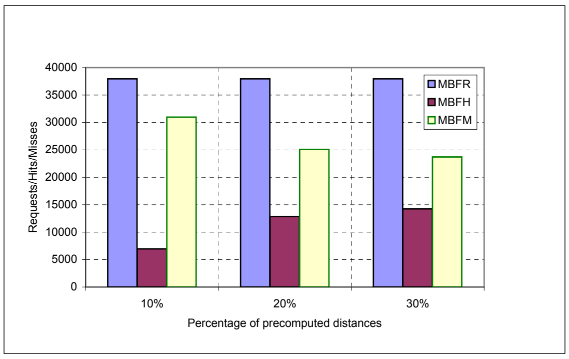


Figure 19: Total running time (in sec) for M-treeII and M-treeI methods.

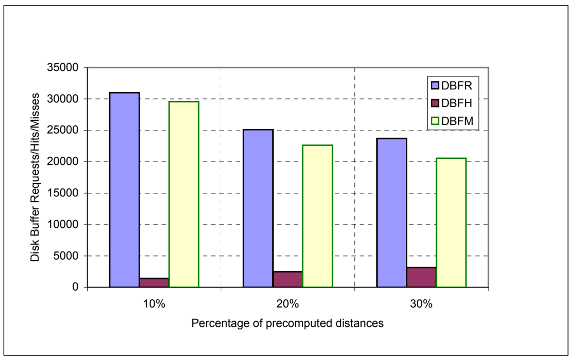
better than the method that utilizes two M-trees (one for D_{net} and one for D_{time}). However, the existence of two M-trees offers a higher degree of flexibility during query processing, since we can search for similar trajectories based: (i) only on network distance D_{net}, (ii) only on time distance D_{time} and (iii) both on network and time distances D_{net} and D_{time}. Moreover, different clustering schemes can be applied. More specifically, using the two separate M-trees, a clustering algorithm can provide clusters for D_{net} or D_{time}. Finally, more choices for query optimization are available if both indexes are utilized, since the query execution engine can form an efficient query execution plan according to the selectivities of the search distances E_{net} and E_{time}, and traverse the M-trees accordingly.

5.6 Impact of Precomputed Distances

The previous experiments have been conducted by having all network distances precomputed and stored on disk. It has been observed that the precomputation reduces the required computational costs during network-based distance calculations. However, the precomputation assumption may not be realistic in very large spatial networks containing many thousands of nodes. However,

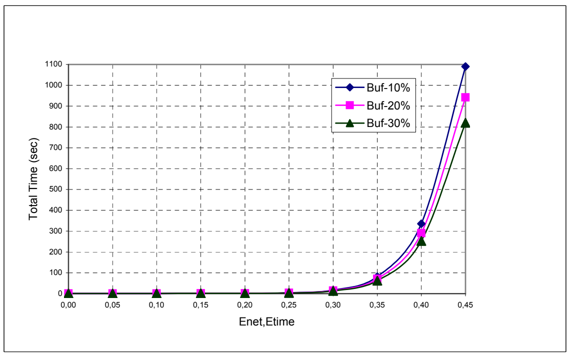


(a) Memory buffer activity

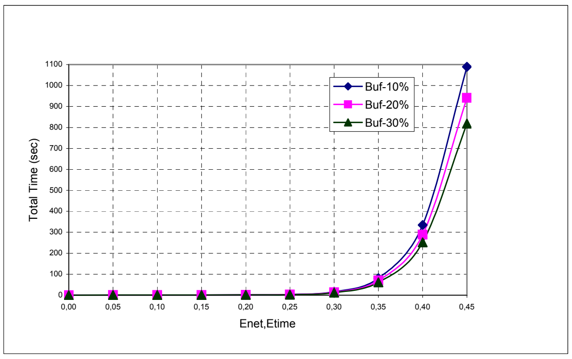


(b) Disk buffer activity

Figure 20: Memory and disk buffer activity for variable disk buffer sizes.



(a) M-treeII



(b) M-treeI

Figure 21: Total running time for D_{net2+1} for variable query radius and disk buffer sizes.

even for small spatial networks, if the main memory buffer fails to achieve an acceptable hit ratio, many distance computations will be invoked, resulting in performance degradation.

Figures 20 and 21 show some interesting results regarding the performance of trajectory similarity queries, when only a subset of the total distances are precomputed. The performance of D_{net1} measure is illustrated in Figure 20, which depicts the activity of the memory-based (a) as well as the disk-based buffer (b). It is evident that by increasing the number of precomputed distances the total running time of trajectory similarity queries decreases but the cost is still significant, raising problems for ad-hoc query processing. On the other hand, the use of the Euclidean distance for filtering purposes results in a much more efficient scheme, as it is illustrated in Figure 21.

6 Conclusions

Although there is significant research work performed on trajectory similarity on moving objects trajectories, the vast majority of the proposed approaches assume that objects can move freely without any motion restrictions. In this paper, we have studied the problem of trajectory similarity query processing in network-constrained moving objects. We have defined two concepts of similarity. The first is based on the network distance and the second is based on the time characteristics of the trajectories. By using these concepts, we have defined distance measures D_{net} to capture the network similarity and a distance measure D_{time} to capture the time-based similarity of trajectories. All proposed measures satisfy the metric space properties, and therefore, metric-based access methods can be used for efficient indexing and searching.

To support trajectories of different description lengths, a decomposition process is applied. Each trajectory is split to a number of sub-trajectories, which are then indexed by M-trees. The NET-M-tree is used for the D_{net} measure, whereas the TIME-M-tree is used for the D_{time} measure. Two methods have been studied: (i) the M-treeI method, which uses only the NET-M-tree and (ii) the M-treeII method, which utilizes both trees. Performance evaluation results show that trajectory similarity can be efficiently supported by these schemes. In all the experiments conducted, the method that uses only one M-tree performs marginally better than the method which utilizes two M-trees. However, the existence of two M-trees offers a higher degree of flexibility during query processing.

Future research may involve: (i) the investigation of alternative indexing schemes, (ii) the study of approximate processing and (iii) the efficient support of trajectory-based k-nearest-neighbor processing, and (iv) the utilization of the proposed similarity measures for data mining (e.g., trajectory clustering).

References

[1] T. Brinkhoff: “A Framework for Generating Network-Based Moving Objects”, *Geoinformatica*, Vol6, No.2, pp.153-180, 2002.

[2] P. Ciaccia, M. Patella, P. Zezula: “M-tree: An Efficient Access Method for Similarity Search in Metric Spaces”, *Proceedings of the 23rd International Conference on Very Large Databases (VLDB)*, 1997.

[3] C. Faloutsos, M. Ranganathan, Y. Manolopoulos: “Fast Subsequence Matching in Time-Series Databases”, *Proceedings of the ACM SIGMOD Conference*, 1994.

[4] A. Guttman: “R-trees: a dynamic index structure for spatial searching”, *Proceedings of the ACM SIGMOD Conference*, pp.4757, 1984.

[5] J.-R. Hwang, H.-Y. Kang and K.-J. Li: “Spatio-temporal Similarity Analysis Between Trajectories on Road Networks”, *ER Workshops*, pp.280-289, 2005.

[6] J.-R. Hwang, H.-Y. Kang, K.-J. Li: “Searching for Similar Trajectories on Road Networks Using Spatio-temporal Similarity”, *Proceedings of the 10th East European Conference on Advances in Databases and Information Systems (ADBIS)*, pp.282-295, 2006.

[7] C.S. Jensen, J. Kolarvr, T.B. Pedersen, I. Timko: “Nearest Neighbor Queries in Road Networks”, *Proceedings of the 11th ACM International Symposium on Advances in Geographic Information Systems (ACM GIS)*, 2003.

[8] G. Kollios, D. Gounopoulos and V.J. Tsotras: “Nearest Neighbor Queries in a Mobile Environment”, *Proceedings of the International Workshop on Spatio-temporal Database Management*, pp.119-134, 1999.

[9] G. Kollios, D. Gunopoulos, V. Tsotras: "On Indexing Mobile Objects", *ACM PODS*, pp.261-272, 1999.

[10] I. Lazaridis, K. Porkaew, S. Mehrotra: "Dynamic Queries over Mobile Objects", *EDBT*, pp.269-286, 2002.

[11] P. Laurinen , P. Siirtola , J. Roning: "Efficient Algorithm for Calculating Similarity Between Trajectories Containing an Increasing Dimension", *Proceedings of the 24th IASTED International Conference on Artificial Intelligence and Applications*, pp.392-399, 2006.

[12] S.-L.Lee, S.-J. Chun, D.-H. Kim, J.-H. Lee, C.-W. Chung: "Similarity Search for Multidimensional Data Sequences", *Proceedings of the 16th International Conference on Data Engineering (ICDE)*, 2000.

[13] D. Lomet, B. Salsberg: "Access Methods for Multiversion Data", *ACM SIGMOD*, pp.315-324, 1989.

[14] N. Meratnia , R. A. de By: "Aggregation and Comparison of Trajectories", *Proceedings of the 10th ACM international symposium on Advances in Geographic Information Systems (ACM GIS)*, 2002.

[15] M.A. Nascimento, J.R.O. Silva: "Towards Historical R-trees", *ACM SAC*, 1998.

[16] D. Papadias, J. Zhang, N. Mamoulis: "Query Processing in Spatial Network Databases", *Proceedings of the 29th International Conference on Very Large Databases (VLDB)*, 2003.

[17] D. Pfoser, C.S. Jensen, Y. Theodoridis: "Novel Approaches to the Indexing of Moving Object Trajectories", *Proceedings of the 26th International Conference on Very Large Databases (VLDB)*, pp.395-406, 2000.

[18] S. Saltenis, C.S. Jensen, S. Leutenegger, M. Lopez: "Indexing the Positions of Continuously Moving Objects", *ACM SIGMOD*, pp.331-342, 2000.

[19] J. Sankaranarayanan, H. Alborzi, H. Samet: "Efficient Query Processing on Spatial Networks", *Proceedings of the 13th ACM International Symposium on Geographic Information Systems (ACM GIS)*, 2005.

[20] Y. Tao, D. Papadias: "Efficient Historical R-trees", *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, 2001.

[21] Y. Tao and D. Papadias: "MV3R-tree - a Spatio-Temporal Access Method for Timestamp and Interval Queries", *Proceedings of the 27th International Conference on Very Large Databases (VLDB*, pp.431- 440, 2001.

[22] Y. Theodoridis, T. Sellis, A.N. Papadopoulos and Y. Manolopoulos: "Specifications for Efficient Indexing in Spatio-temporal Databases", *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, 1998.

[23] C. Traina, A.J.M. Traina, B. Seeger and C. Faloutsos: "Slim-Trees: High Performance Metric Trees Minimizing Overlap Between Nodes", *Proceedings of the 7th International Conference on Extending Database Technology (EDBT)*, pp.51-65, 2000.

[24] M. Vlachos, D. Gunopulos, G. Kollios: "Robust Similarity Measures for Mobile Object Trajectories", *Proceedings of the 5th International Workshop on Mobility in Databases and Distributed Systems*, 2002.

[25] M. Vlachos, G. Kollios, D. Gunopulos: "Discovering Similar Multidimensional Trajectories", *Proceedings of the 18th IEEE International Conference on Data Engineering (ICDE)*, 2002.

[26] O. Wolfson, B. Xu, S. Chamberlain, L. Jiang: "Moving Objects Databases: Issues and Solutions", *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*, pp.111-122, 1998.

[27] O. Wolfson, B. Xu, S. Chamberlain: "Location Prediction and Queries for Tracking Moving Objects", *Proceedings of the 16th IEEE International Conference on Data Engineering (ICDE)*, pp.687-688, 2000.

[28] Y. Yanagisawa, J.-I. Akahani, T. Satoh: "Shape-Based Similarity Query for Trajectory of Mobile Objects", *Proceedings of the 4th International Conference on Mobile Data Management (MDM)*, pp.63-77, 2003.

[29] J.S. Yoo, S. Shekhar: "In-Route Nearest Neighbor Queries", *GeoInformatica*, Vol.9, No.2, pp.117-137, 2005.