Trajectory Similarity Search in Spatial Networks

Eleftherios Tiakas Apostolos N. Papadopoulos Alexandros Nanopoulos Yannis 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 are assumed to move on predefined spatial networks such as road segments, railways, invisible air routes. Moving objects may exhibit similarity with respect to their traversed paths, and therefore two objects can be correlated based on their path similarity. In this paper, we study similarity search for moving object trajectories for spatial networks. The problem poses some important challenges, since it is quite different from the case where objects are allowed to move without any motion restrictions. Experimental results performed on real-life spatial networks show that trajectory similarity can be supported in an effective and efficient manner, by using metric-based access methods.

1 Introduction

In location-based services it is important to pose queries based on the objects' 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 (spatiotemporal queries). Evidently, indexing schemes are ubiquitous to efficiently support queries on moving objects, by quickly discarding parts of the database which are not relevant.

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. By identifying similar trajectories, effective data mining techniques (e.g., clustering) can be applied to discover useful patterns. The majority of the research proposals investigating trajectory similarity are based on the assumption that objects can move freely without restrictions on their motion. In this paper, we focus on trajectory similarity of moving objects by considering that object mobility is constrained by an underlying spatial network.

The rest of the article is organized as follows. In the next section we give the appropriate background, present related work performed and explain the motivation and contribution. 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 and Contribution

Although query processing for location-based services is an active area of research, the issue of trajectory similarity has not received the required attention. Moreover, the issue of trajectory similarity on spatial networks has been touched only recently. In this section, we give the appropriate background on spatial networks and trajectory similarity, and motivated contributions.

2.1 Spatial Networks

In several applications, the mobility of objects is constrained by an underlying spatial network. This means that objects can not move freely, and their position must satisfy the network constraints. Network connectivity is usually modeled by using a graph representation, composed by a set of vertices (nodes) and a set of edges (connections). Depending on the application the graph may be *weighted* (a

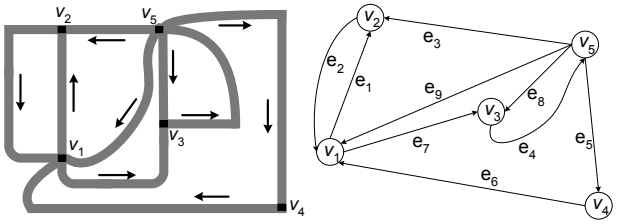


Figure 1. A road network and its graph.

cost is assigned to each edge) and *directed* (each edge has an orientation). Figure 1 illustrates an example of a spatial network corresponding to a part of a city road network, and its graph representation.

Several research efforts have been performed towards efficient spatial and spatio-temporal query processing in spatial networks. In [9] nearest-neighbor query processing is achieved by using a mapping technique. Nearest-neighbor queries in road networks have been also studied in [6]. In [8], the authors study query processing for stationary datasets, 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 [10], where given a trajectory source and destination the smallest detour is calculated.

2.2 Trajectory Similarity

An interesting research issue is to determine a way of expressing the similarity among trajectories of moving objects. 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 similarity 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 0. On the other hand, if the two trajectories are dissimilar, we expect the value $D(T_a, T_b)$ to be close to 1.

In several research proposals, trajectory similarity is viewed as the multidimensional counterpart of time series similarity. In [7] 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.

Another approach for expressing the similarity between two trajectories is studied in [12, 13]. A more robust distance metric is used, based on the *longest common subsequence* (LCS) between two trajectories. This metric is more immune to noise than the Minkowski distance. Because the proposed distance does not satisfy the metric space properties, indexing is achieved by utilizing the index structure of a hierarchical clustering algorithm.

To the best of the authors' knowledge, the only research work studying trajectory similarity on networks is the work in [5]. The authors propose a simple similarity measure based on POIs (points of interest). No details are given with respect to the access methods required to provide efficient similarity search. Moreover, it is not clear how trajectories of different lengths are handled, and finally, no discussion is performed regarding the metric space properties of the proposed distance measures.

2.3 Motivation and Contribution

The research work performed for trajectory similarity assumes that objects can move anywhere in the 2-D or 3-D Euclidean space, without any restrictions in their motion. Taking into consideration that a large number of applications require that objects are embedded to an underlying network, the challenge is to express trajectory similarity by respecting the network constraints.

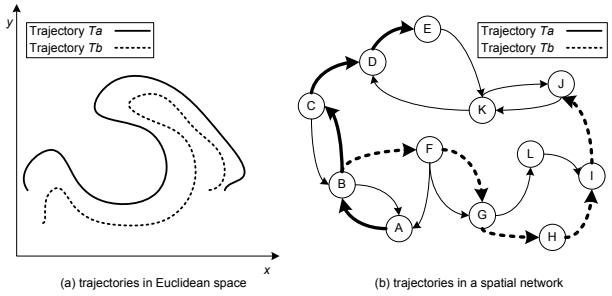


Figure 2. Trajectories in (a) 2-D Euclidean space, and (b) in a spatial network.

Figure 2 illustrates an example that shows the differences between restricted and unrestricted trajectories. Objects moving in a spatial network follow specific paths determined by the graph topology, and therefore arbitrary movement is prohibited.

The motivation behind our work is summarized as follows:

(I) 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.

(II) The method proposed in [12, 13] employs a similarity distance based on the longest common subsequence between two trajectories. This approach leads to a distance measure which 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 can not 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.

In the next section we study in detail the proposed similarity model for trajectory similarity search in spatial networks aiming at: (i) the definition of a similarity distance between trajectories that satisfies 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.

3 Trajectory Similarity Measures

Let $\mathcal{T}$ be the 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 is defined as:

$$T = ((v_1, t_1), (v_2, t_2), ..., (v_m, t_m)) \quad (1)$$

where m is the trajectory 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 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 frequently for the rest of the study.

3.1 Expressing Trajectory Similarity

We begin our exploration by assuming that any two trajectories have the same length. Later, we are going to relax this assumption. Let T_a and T_b be two trajectories, each of 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 $\forall i$, $v_{ai} \in \mathcal{V}$ and $v_{bi} \in \mathcal{V}$. Note that, to characterize two trajectories *similar* it is not necessary to share common nodes. Therefore, the similarity measure must take into account the *proximity* of the trajectories.

Symbol	Description
$\mathcal{T}$	set of trajectories
T, T_a, T_b	trajectories
T_q	a query trajectory
m	trajectory length
$\mathcal{V}$	set of graph nodes (vertices)
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 at the i-th node
$d(v_i, v_j)$	network distance between nodes v_i and v_j
$D_{net}(T_a, T_b)$	network distance between T_a and T_b
$D_{time}(T_a, T_b)$	time distance between T_a and T_b
E_{net}	query radius for D_{net}
E_{time}	query radius for D_{time}

Table 1. Symbols used throughout the study.

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. Let $c(v_i, v_j)$ denote the cost to travel from a source node v_i to a destination node v_j. This cost may express the travel distance between the two nodes, the average time required to travel from v_i to v_j, or any other meaningful cost measure which is usually application dependent.

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

$$d(v_i, v_j) = \begin{cases} 0,\ c(v_i, v_j) = 0 \wedge c(v_j, v_i) = 0 \\ \frac{min\{c(v_i,v_j), c(v_j,v_i)\}}{max\{c(v_i,v_j), c(v_j,v_i)\}},\ \text{otherwise} \end{cases} \quad (2)$$

Definition 2
The network distance $D_{net}(T_a, T_b)$ between two trajectories T_a, T_b with length m is defined as:

$$D_{net}(T_a, T_b) = \frac{1}{m} \cdot \sum_{i=1}^{m} (d(v_{ai}, v_{bi})) \quad (3)$$

Proposition 1
The distance measure $d(v_i, v_j)$ satisfies the metric space properties.

Proof
Clearly $d(v_i, v_j) \geq 0$, and $d(v_i, v_j) = d(v_j, v_i)$, therefore the first two metric space properties are hold. It suffices to prove that triangular inequality is satisfied. Let v_x be any other node in the graph representation of the spatial network. Then, we need to prove that

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

and by substitution we get:

$$\frac{min\{c(v_i, v_j), c(v_j, v_i)\}}{max\{c(v_i, v_j), c(v_j, v_i)\}} \leq \frac{min\{c(v_i, v_x), c(v_x, v_i)\}}{max\{c(v_i, v_x), c(v_x, v_i)\}} + \frac{min\{c(v_x, v_j), c(v_j, v_x)\}}{max\{c(v_x, v_j), c(v_j, v_x)\}}$$

The application of min and max functions results in a non-directed weighted graph, where the triangular inequality holds. Therefore, the following inequalities are true:

$$min\{c(v_i, v_j), c(v_j, v_i)\} \leq min\{c(v_i, v_x), c(v_x, v_i)\} + min\{c(v_x, v_j), c(v_j, v_x)\} \quad (4)$$

$$max\{c(v_i, v_j), c(v_j, v_i)\} \leq max\{c(v_i, v_x), c(v_x, v_i)\} + max\{c(v_x, v_j), c(v_j, v_x)\} \quad (5)$$

For simplicity, we use the following substitutions:

$$\begin{aligned} C_1 &= min\{c(v_i, v_j), c(v_j, v_i)\} \\ C_2 &= min\{c(v_i, v_x), c(v_x, v_i)\} \\ C_3 &= min\{c(v_x, v_j), c(v_j, v_x)\} \\ C_4 &= max\{c(v_i, v_j), c(v_j, v_i)\} \\ C_5 &= max\{c(v_i, v_x), c(v_x, v_i)\} \\ C_6 &= max\{c(v_x, v_j), c(v_j, v_x)\} \end{aligned}$$

By substitution in Equations 4 and 5 we get:

$$C_1 \leq C_2 + C_3 \quad (6)$$

$$C_4 \leq C_5 + C_6 \quad (7)$$

Since $C_1 \leq C_4$ and $(C_2 + C_3) \leq (C_5 + C_6)$ we can safely divide (6) and (7) above without changing the direction of the inequalities:

$$\frac{C_1}{C_4} \leq \frac{C_2}{C_5 + C_6} + \frac{C_3}{C_5 + C_6} \Rightarrow \frac{C_1}{C_4} \leq \frac{C_2}{C_5} + \frac{C_3}{C_6} \quad (8)$$

□

Proposition 2
The distance $D_{net}(T_a, T_b)$ satisfies the metric space properties.

Proof
Again, the first two metric space properties are satisfied. Therefore, it is sufficient to prove that triangular inequality also holds in this case. Since for any graph nodes v_i, v_j, the distance $d(v_i, v_j)$ satisfies the metric space properties, the summation preserves the inequality direction. Therefore, the triangular inequality holds. □

3.2 Incorporating Time Information

The similarity measure defined in the previous section takes 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 very important.

Definition 3
Given two trajectories $T_a \in \mathcal{T}$ and $T_b \in \mathcal{T}$ with length m, their distance with respect to time is denoted as $D_{time}(T_a, T_b)$ and it is given by:

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

Essentially, the time similarity between two trajectories, as it has been defined in Definition 3, measures their resemblance with respect to the time required to travel from one node to the next. Clearly, the D_{time} measure assumes values in the interval $[0, 1]$, and as the following proposition explains, respects the metric space properties.

Proposition 3
The distance measure $D_{time}(T_a, T_b)$ satisfies the metric space properties.

Proof
By following a similar approach with that used for Propositions 1 and 2. □

We have at hand two different similarity measures D_{net} and D_{time} that can be used to compare trajectories of the same length. As it is mentioned, several applications may require both distance measures to extract useful knowledge. Therefore, one alternative is to combine these two measures 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) similarity can then be expressed as follows:

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

It is evident that the distance metric 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.

Another alternative is to use D_{net} and D_{time} separately. This way, two distances are required to be posed by the query. The distance E_{net} expresses the desired similarity with respect to the D_{net} metric, whereas the distance E_{time} expresses the desired similarity regarding the D_{time} metric. If the user wishes to focus only on the network distance D_{net}, 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 accordingly.

4 Indexing and Query Processing

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

Let T be a trajectory of 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 3 illustrates an example of the decomposition process, where m=6 and μ=3.

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

Our next step is to index the set $\mathcal{S}$ of sub-trajectories, enabling efficient query processing. Towards this direction,

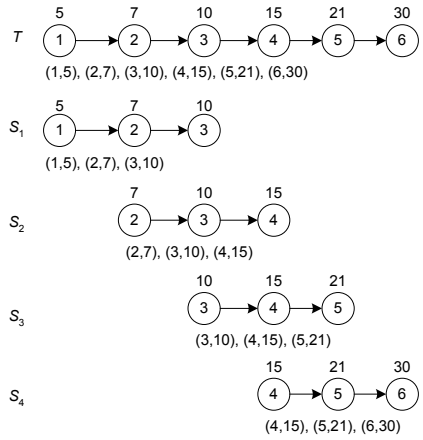


Figure 3. Decomposition (m=6, μ=3).

we propose two schemes, which are both based on the Mtree access method. 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 Mtree 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 Mtree 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 Mtrees. Note that, although we use Mtrees, any metric access method can been applied (e.g., SlimTrees [11]). Two alternatives are followed towards indexing sub-trajectories:

MtreeI method: In this scheme, only the NET-Mtree 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 Mtree is used.

MtreeII method: In this scheme, two Mtrees are used to handle D_{net} and D_{time} separately. These trees are termed NET-Mtree and TIME-Mtree respectively. Each Mtree 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.

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 length μ. If this is not true, padding is performed by repeating, for example, the last node of the trajectory several times, until the length μ is reached. In the general case where the length of T_q is greater than μ, the decomposition

process is applied to obtain the sub-trajectories of T_q.

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 MtreeI or MtreeII as they have been described previously. A trajectory is part of the answer if there is at least one of its sub-trajectories that satisfies the network and time constraints for at least one query sub-trajectory.

5 Performance Evaluation

The proposed approach has been implemented in C++ and the experiments have been conducted on a Pentium IV with 1GB RAM running Windows XP. To generate trajectories the generator of [1] has been used. The data set we have used for the experiments consists of 3,797 trajectories of objects moving on the roads of Oldenburg city. Each trajectory has a length of at least 10. A sliding window of length μ = 10 has been used to generate the sub-trajectories of each trajectory. Therefore, the total number of sub-trajectories produced is 75,144.

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

---

1. **if** length(T_q) $< \mu$
2. pad T_q with the last trajectory node
3. $QS(T_q) \leftarrow T_q$
4. **else**
5. $QS(T_q) \leftarrow$ all sub-trajectories of T_q of length μ
6. **end if**
7. **for each** query sub-trajectory $qs \in QS(T_q)$
8. **if** method MtreeI is used
9. search NET-Mtree using qs and E_{net}
10. $C_{net} \leftarrow$ candidate sub-trajectories from NET-Mtree
11. check every sub-trajectory in C against E_{time}
12. update $\mathcal{AS}$
13. **else if** method MtreeII is used
14. search NET-Mtree using qs and E_{net}
15. $C_{net} \leftarrow$ candidate sub-trajectories from NET-Mtree
16. search TIME-Mtree using qs and E_{time}
17. $C_{time} \leftarrow$ candidate sub-trajectories from TIME-Mtree
18. $\mathcal{AS} \leftarrow C_{net} \cap C_{time}$
19. **end if**
20. **end for**
21. calculate $\mathcal{AT}$ from $\mathcal{AS}$
22. return ($\mathcal{AS}$, $\mathcal{AT}$)

Figure 4. Similarity search algorithm.

For the MtreeI method only one Mtree is built based on the D_{net} measure, whereas for the MtreeII method two Mtrees are built based on D_{net} and D_{time}. The NET-Mtree which is implemented based on the D_{net} metric and the TIME-Mtree implemented based on the D_{time} metric. Recall that, both trees handle sub-trajectories of length μ=10 and not complete trajectories of moving objects.

The distance between two nodes of the graph corresponds to the shortest path distance between these nodes. The number of vertices in the Oldenburg data set is 6,105. Therefore, the total number of pre-computed distances among all possible pairs of vertices is 37,271,025. An LRU buffer has been used to keep a number of distances in main memory. The rest are kept in a hash-based file on disk. The size of the buffer for the main memory distances has been set to 2,000, which is a relatively small value compared to the total number of pair-wise distances. Figure 5 depicts the number of network distance calculation requests, the number of hits and the number of misses. It is evident, that about 25% 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. We note that this buffer is used only for the search in the NET-Mtree. The calculation of D_{time} does not require any network distance calculation.

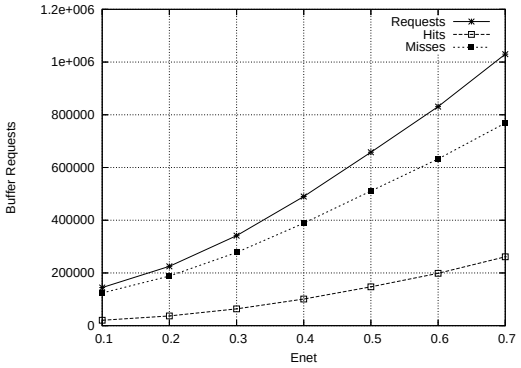
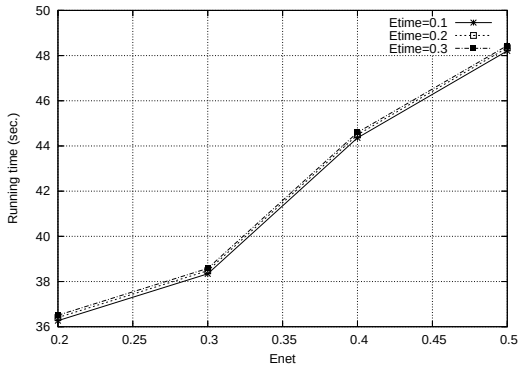


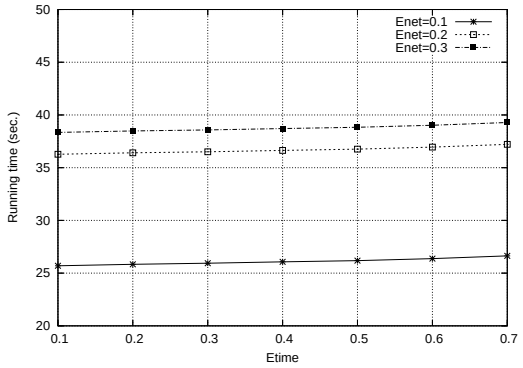
Figure 5. Distance requests, hits and misses.

Figure 6(a) depicts the performance of the MtreeII method for different values of the parameter E_{net}. Three curves are shown, for three different values of the E_{time} parameter. It is evident, that by increasing E_{net} the total processing time also increases. Moreover, we observe that E_{time} does not have a serious impact in query processing performance.

Similar results are obtained from Figure 6(b), which depicts the query processing time for different values of the parameter E_{time}. Again, there are three curves in the graph for three different values of E_{net}. It is evident that by increasing E_{time} the overall query performance is not af-



(a) variable E_{net}



(b) variable E_{time}

Figure 6. Query response time vs (a) E_{net} and (b) E_{time}.

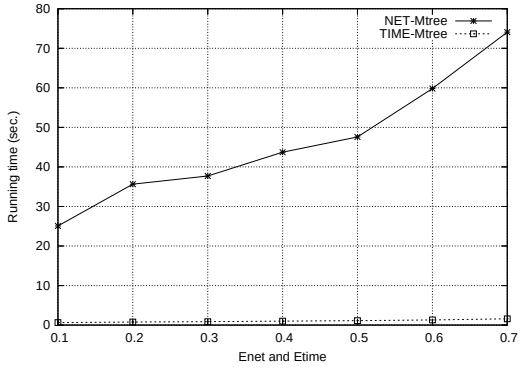


Figure 7. Time required to search the two Mtrees in the MtreeII method.

fected significantly. Therefore, the overall query processing cost is dominated by the network-based similarity search. This is clearly shown in Figure 7, which illustrates the time required to search the NET-Mtree and the TIME-Mtree access methods when E_{net} and E_{time} assumes the same value between 0 and 1. We obtained similar results for the MtreeI method, which uses only the NET-Mtree. Again, the dominant part of the query processing is the determination of network distances.

In the MtreeI approach, the number of D_{time} calculations is determined by the number of candidate subtrajectories return by the search in the NET-Mtree. On the other hand, when the TIME-Mtree is used, the number of D_{time} calculations are affected by the parameter E_{time}. Figure 8 illustrates the number of D_{time} calculations performed by MtreeI and MtreeII methods.

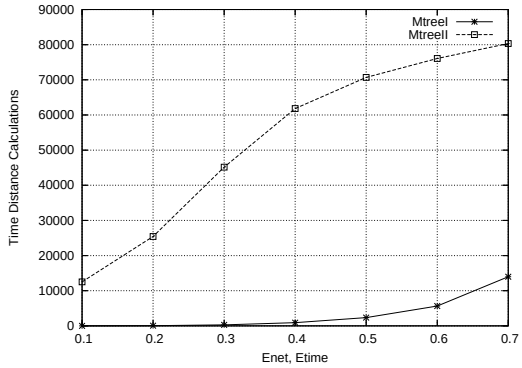


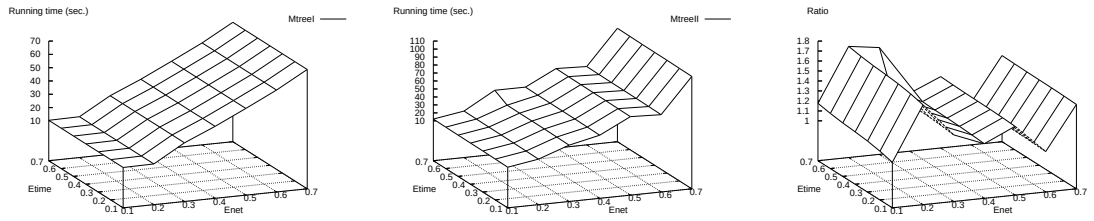
Figure 8. Number of D_{time} calculations.

The total processing costs for the two methods is depicted in Figure 9. The cost of MtreeI and MtreeII is shown in Figure 9(a) and (b) respectively, whereas in (c) we give the total cost ratio of MtreeII over MtreeI. Evidently, query processing with the MtreeI method is more efficient than that of the MtreeII method, for all combinations of several values of E_{net} and E_{time} parameters.

In all the experiments conducted, the method that uses only one Mtree on the D_{net} metric performs better than the method which utilizes two Mtrees (one for D_{net} and one for D_{time}). However, the existence of two Mtrees offer a higher degree of flexibility during query processing, since we can search for similar trajectories based: (i) only on D_{net}, (ii) only on D_{time} and (iii) both on D_{net} and D_{time}. Moreover, different clustering schemes can be achieved based on the two metrics. More specifically, using the two separate Mtrees, 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 Mtrees accordingly.

6 Concluding Remarks

Although there is significant research work performed on trajectory similarity on moving objects trajectories, the



(a) MtreeI (b) MtreeII (c) Ratio MtreeII over MtreeI

Figure 9. Comparison of total running time.

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 two similarity measures, D_{net} to capture the network distance and D_{time} to capture the time-based distance of trajectories. Both measures satisfy the metric space properties, and therefore, metric-based access methods can be used for efficient indexing. Performance evaluation results have shown that trajectory similarity can be efficiently supported by these schemes, provided that node distances are pre-computed.

Acknowledgments

Research supported by the 2003-2005 Serbian-Greek joint research program, funded by the General Secretariat of Research and Technology (GSRT), Ministry of Development, Greece.

References

[1] T. Brinhoff, "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", *23rd International Conference on Very Large Databases*, 1997.

[3] C. Faloutsos, M. Ranganathan, Y. Manolopoulos, "Fast Subsequence Matching in Time-Series Databases", *ACM SIGMOD Conference*, 1994.

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

[5] J.-R. Hwang, H.-Y. Kang, K.-J. Li, "Spatiotemporal Similarity Analysis Between Trajectories on Road Networks", *ER Workshops*, pp.280-289, 2005.

[6] C.S. Jensen, J. Kolarvr, T.B. Pedersen, I. Timko, "Nearest Neighbor Queries in Road Networks", *11th ACM International Symposium on Advances in Geographic Information Systems*, 2003

[7] S.-L.Lee, S.-J. Chun, D.-H. Kim, J.-H. Lee, C.-W. Chung, "Similarity Search for Multidimensional Data Sequences", *16th International Conference on Data Engineering*, 2000.

[8] D. Papadias, J. Zhang, N. Mamoulis, "Query Processing in Spatial Network Databases", *29th International Conference on Very Large Databases*, 2003.

[9] J. Sankaranarayanan, H. Alborzi, H. Samet, "Efficient Query Processing on Spatial Networks", *13th ACM International Symposium on Geographic Information Systems*, 2005.

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

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

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

[13] M. Vlachos, G. Kollios, D. Gunopulos, "Discovering Similar Multidimensional Trajectories", *18th IEEE International Conference on Data Engineering*, 2002.