Summarizing Trajectories into k-Primary Corridors: A Summary of Results

Michael R. Evans, Dev Oliver
Shashi Shekhar
Computer Science and Engineering
University of Minnesota
Minneapolis, MN, USA
{mevans,oliver,shekhar}@cs.umn.edu

Francis Harvey
Geography
University of Minnesota
Minneapolis, MN, USA
fharvey@umn.edu

ABSTRACT

Given a set of GPS trajectories on a road network, the goal of the k-Primary Corridors (k-PC) problem is to summarize trajectories into k groups, each represented by its most central trajectory. This problem is important to a variety of domains, such as transportation services interested in finding primary corridors for public transportation or greener travel (e.g., bicycling) by leveraging emerging GPS trajectory datasets. Related trajectory mining approaches, e.g., density or frequency based hot-routes, focus on anomaly detection rather than summarization and may not be effective for the k-PC problem. The k-PC problem is challenging due to the computational cost of creating the track similarity matrix. A naïve graph-based approach to compute a single element of this track similarity matrix requires multiple invocations of common shortest-path algorithms (e.g., Dijkstra). To reduce the computational cost of creating this track similarity matrix, we propose a novel algorithm that switches from a graph-based view to a matrix-based view, computing each element in the matrix with a single invocation of a shortest-path algorithm. Experimental results show that these ideas substantially reduce computational cost without altering the results.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications–(Spatial Databases and GIS)

General Terms

Algorithms, Performance, Experimentation

Keywords

Trajectory Summarization, Spatial Data Mining, GPS

1. INTRODUCTION

Problem: Given a set of trajectories on a road network, the goal of the k-Primary Corridors problem is to summarize trajectories into k groups, each represented by its most central trajectory. Figure 1(a) shows a real-world GPS dataset of a number of trips taken by bicyclists in Minneapolis, MN. The darkness indicates the usage levels of each road segment. The computational problem is summarizing this set of trajectories into a set of k-Primary Corridors. One potential solution is shown in Figure 1(b) for k=8. Each identified primary corridor represents a subset of bike tracks from the original dataset. Note that the output of the k-PC problem is distinct from that of hot or frequent routes, as it is a summary of all the given trajectories partitioned into k primary corridors.

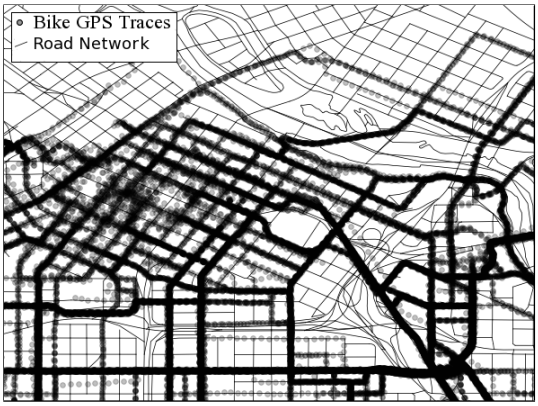
Motivation: The k-Primary Corridor problem is important due to a number of societal applications, such as city-wide bus route modification or bicycle corridor selection, among other urban development applications. Let us consider the problem of determining primary bicycle corridors through a city to facilitate safe and efficient bicycle travel. By selecting representative trajectories for a given group of commuters, the overall alteration to commuters routes is minimized, encouraging use. Facilitating commuter bicycle traffic has shown in the past to have numerous societal benefits, such as reduced greenhouse gas emissions and healthcare costs [11].

Related Work: Graph-based trajectory clustering in existing literature is focused on 'hot' trajectories, e.g., popular tracks found via different approaches (e.g., trajectory pattern mining [4, 13, 15], computational geometry [1], frequent subgraphs [8], density clustering [2,7,9,10], hierarchical clustering [5, 12, 14]). Note that these 'hot' route techniques traditionally focus on track anomalies (e.g., high-frequency tracks), as compared to summarizing the dataset as a whole.

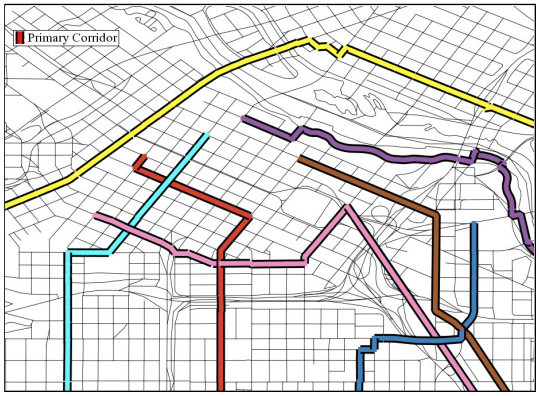
To address these limitations, our work focuses on centroid-based (prototype) clustering, namely the summarization of a group of objects represented by a set of centroid objects. Each primary corridor we identify can be considered a representative track of a cluster of tracks, where the actual track identified attempts to minimize the overall distance to each other track within its cluster (as compared to finding 'hot', or frequently traveled tracks).

Challenges: Summarizing tracks into k-primary corridors is challenging primarily due to the extensive number of shortest-path network distance calculations needed to compute similarities between tracks (the track similarity matrix (TSM)). For example, given two tracks consisting of 10 nodes each, a naïve graph-based algorithm to measure similarity would need to calculate the distance of each node to

ACM SIGSPATIAL GIS '12, November, 6-9 2012. Redondo Beach, CA, USA



(a) Recorded GPS points from bicyclists in Minneapolis, MN. Intensity indicates number of points.



(b) Set of 8-Primary Corridors identified from Bike GPS traces in Minneapolis, MN. (Best in color)

Figure 1: Example Input and Output of the k-Primary Corridor problem.

the closest node in the other track via a shortest-path algorithm, e.g., Dijkstra. In the worst case, that will require at least 10 shortest-path distance calculations to find the nearest node in the other track (one for each node). Doing this for both tracks would result in 90 shortest-path computations (10 * 10 - 10 self joins) on a directed graph, 45 on an undirected graph.

Table 1: Runtime comparison of steps in the k-Primary Corridor problem (100 nodes/tracks, k=5)

Steps of k-PC	Runtime	Percentage
Track Similarity Matrix	46.56 sec	94.1%
Partitioning / Clustering	2.96 sec	5.9%
Total	49.52 sec	100%

Contributions: This paper makes the following contribution claims: (1) we introduce the k-Primary Corridor (k-PC) problem, (2), we propose two algorithms for computing the track similarity matrix along with an algorithm for computing k-Primary Corridors, and (3) experimental evaluation of the proposed algorithms.

2. PROBLEM FORMULATION

In this section, we describe the basic concepts required to describe the k-primary corridor problem. We provide an example dataset and finally a formal problem statement.

2.1 Basic Concepts

We begin with a review of digital road maps and GPS tracks, and then introduce primary corridors and our proposed track similarity measure.

A **road network** can be represented as a graph, with nodes representing road intersections and edges representing the segments of roads connecting adjacent road intersections. Edges in these graphs can have weights, or distance values, indicating the length of the road segment.

A **track** in this paper represents a trajectory of a bicyclist across a road network. The original GPS trace of the individual is pre-processed to be map-matched into a series of nodes and edges creating a track through the road network graph. Figure 2 contains three tracks, color-coded as shown in the Legend.

Network distance is the sum of the length of edges required to traverse the graph between two specific nodes. A number of well-known algorithms compute shortest-path network distance, for simplicity we will use the commonly-used Dijkstra algorithm [3].

To measure the similarity of two tracks, we define a **track similarity** function, in essence measuring the average minimum distance from any node in one track to any node in the other track. When computing the overall average minimum distance, we use the following equation:

$$s(t_i, t_j) = \frac{1}{|t_i|} \sum_{n \in t_i} \min_{m \in t_j} (ShortestPath(n, m))$$

In this formulation, the similarity measure is computing the average minimum distance from any node in Track i to any node in Track j. Variables n and m represent nodes in the input tracks i and j, respectively.

To compute the track similarity of Track 1 to Track 3, we first find the distance from each node in Track 1 to the closest node in Track 3. We then sum those distances and normalize them by the length (number of nodes) of Track 1, resulting in a track similarity score of 3.

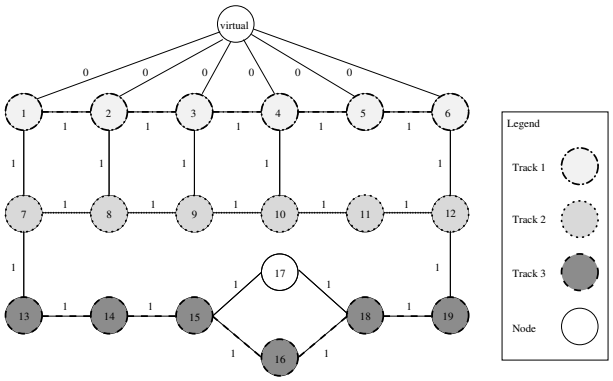


Figure 2: Road network represented as an undirected graph with three tracks.

We define a **primary corridor** representative as a track that summarizes a group of other tracks via a track similarity function. The chosen track minimizes the distance from all other tracks within it's represented group. These primary corridors are meant to semantically represent key paths through the road network to summarizing the overall dataset. An example of a primary corridor is Track 2 in Figure 2, as it has the lowest distance from the other tracks.

Table 2: Track Similarity Matrix for Figure 2.

	Track 1	Track 2	Track 3	Row Sum
Track 1	0	1.16	3	4.16
Track 2	1.16	0	2	3.16
Track 3	3	2	0	5

2.2 Problem Statement

The k-Primary Corridor problem can be formulated as follows:

Given:

- Road Network: $G = \{V,E\}$
- Collection of Tracks: T
- Number of Primary Corridors: k

Find:

- k Primary Corridors

Objective:

- $Minimize(\sum_{t \in T}$ track-similarity(t, t's assigned primary corridor representative track))

Constraints:

- Each primary corridor is represented by a track $\in T$
- G is a connected graph with nonnegative edge weights

Example: Using the example dataset in Figure 2, we explain the components of the formal problem statement shown above. The given road network is the set of nodes and edges represented by circles and lines, respectively. Tracks 1, 2, and 3 are the collection of input tracks T. Lastly, for this example, we will choose 1 primary corridor (k=1). We are therefore to find 1 representative track to summarize the entire set T. To do this, we will choose a track that minimizes the intra-cluster pairwise distance between tracks, where the distance is provided in the track similarity matrix shown in Table 2. The choice is Track 2 as illustrated in the last column of Table 2.

3. PROPOSED APPROACH

The k-Primary Corridor problem can be solved via a two-step process. First the track similarity matrix for all pairs of tracks is computed. As described above, this requires measuring the similarity of each pair of tracks in the dataset, and within that, each pair of nodes in each track, resulting in a large number of network distance computations. Once the track similarity matrix has been created, a k-medoid clustering algorithm can be applied to find the k-Primary Corridors. In the naïve approach, we use the well known Partitioning Around Medoids (PAM) [6] algorithm to summarize the tracks and choose representative tracks as medoids based on the similarity matrix we computed in the first step. In this section, we propose a novel algorithm for reducing the computational cost of computing the track similarity matrix, followed by a proposed solution to the k-Primary Corridor problem.

3.1 Computing the Track Similarity Matrix

In this section, we introduce two algorithms for computing the track similarity matrix, the dominant computational expense in the k-Primary Corridor problem. The first retains a graph-based view, requiring pairwise node to node shortest-path network distance calculations within each track. The second uses a matrix-based view, computing elements of the track similarity matrix one at a time, resulting in a single network distance computation per element.

Graph-Node Track Similarity (GNTS): To generate the Track Similarity Matrix M, we compute the network distance between each pair of nodes within each pair of tracks. This naïve Graph-Node Track Similarity (GNTS) algorithm is intuitive, essentially invoking a shortest-path computation for each combination of nodes within each track pair. For example, in Figure 2, to compute the similarity between Track 1 and Track 3, starting from Node 1, a shortest-path invocation would occur for each node in Track 3 (13, 14, 15, 16, 18, 19). This repeats for each node in Track 1 to compute one element in the row of the track similarity matrix $s(t_1, t_3)$. This may be repeated over all ordered pairs of tracks to compute the entire matrix, such as the one in Table 2.

Matrix-Element Track Similarity (METS): While the proposed GNTS approach in the previous section produces the correct output, a careful examination reveals a major computational bottleneck as illustrated in Table 1: multiple invocations of shortest-path algorithms for each element in the track similarity matrix. Using a matrix-based view, we introduce a novel algorithm for computing each element in the track similarity matrix with a single invocation of a shortest-path algorithm called Matrix-Element Track Similarity (METS).

Algorithm 1 Matrix-Element Track Similarity (METS)

Input:
- Road Network $G = \{V, E\}$
- Tracks T: Set of tracks

Output:
- Track Similarity Matrix $M_{T \times T}$

```
1: for all tracks i in T do
2:   for all tracks j in T do
3:     if i ≡ j then
4:       M[i][j] ← 0
5:     else
6:       Create virtual node vSource
7:       G ← vSource
8:       for all nodes n in i do
9:         Create 0-length edge from vSource to n
10:      end for
11:      for all nodes m in j do
12:        sinks ← m
13:      end for
14:      dists[] = Dijkstra(vSource, sinks)
15:      Remove vSource from G
16:      sum = 0
17:      for all nodes m in j do
18:        sum ← sum + dists[m]
19:      end for
20:      M[i][j] ← sum/|i|
21:    end if
22:  end for
23: end for
24: return M_{T×T}
```

Execution Trace of METS: Given a road network G and a set of tracks T, we calculate the Track Similarity Ma-

trix M in Algorithm 1. Using the same dataset as above, Figure 2, we enumerate through each pair of tracks with two for loops shown on Lines 1 and 2. We will describe one pass through this algorithm using Track 1 as i and Track 3 as j. We first attach a virtual node with edge distance values of 0 to all the nodes in Track 1 in Line 9. In Line 12 we are setting the sink (destination) nodes for the Dijkstra computation. We then initiate a Dijkstra single-source distance computation from the virtual node to all the nodes in the *sinks* set (the nodes in Track 3). That returns the distance from each node in Track 3 to the virtual node *vSource*, essentially returning the distance from each node in Track 3 to its closest node in Track 1 (as the distances from the nodes in Track 1 and *vSource* are all 0). This allows us to compute our track similarity metric in Lines 17 - 20. We then do that for all pairs of tracks to create M.

4. EXPERIMENTAL EVALUATION

In this section, we present our experimental validation for the k-Primary Corridor problem and our proposed solution. In this preliminary work, we explored the impact of the *number of nodes* in the road network on the computational cost of the various track similarity algorithms. The experiments in this section were performed on synthetic datasets. We generated the underlying roadmap (planar graph), along with suitable tracks.

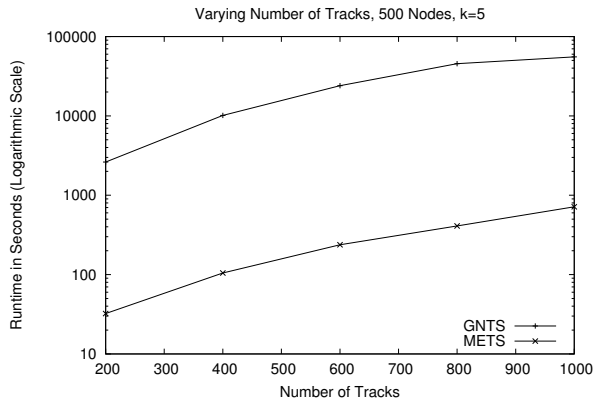


Figure 3: Varying Tracks

In the experiment shown in Figure 3, we created a road network with 500 nodes, an average track length of 25 nodes and a k of 5. The x axis shows the number of tracks given as input, the y axis shows the runtime in seconds on a logarithmic scale. The figure shows all three algorithms varying over the given numbers of tracks, and as is clear, GNTS quickly becomes prohibitive even with this small dataset size.

5. CONCLUSION AND FUTURE WORK

Discovering k-Primary Corridors is important for identifying potential avenues for enhancing pedestrian and bicyclist access, as well as potential applications for determining metro bus corridors and light rail systems. While related work primarily focuses on finding frequently- and densely-traveled routes, the proposed k-primary corridor problem focuses on summarizing the overall dataset, finding primary corridors to represent groups of tracks. Comparing thousands of tracks on large spatial networks is computationally expensive due to a large number of shortest-path network distance computations required to measure similarity between tracks. We proposed a novel matrix-based view and associated algorithms for computing the track similarity matrix, a key bottleneck in the k-Primary Corridor problem. We demonstrated the computational scalability of our algorithms via synthetic dataset experiments. In our future work, we will explore centroid-based corridor generation.

6. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1029711, IIS-CXT IIS-0713214, IGERT DGE-0504195, CRI:IAD CNS-0708604, and USDOD under Grant No. HM1582-08-1-0017, HM1582-07-1-2035, and W9132V-09-C-0009.

7. REFERENCES

[1] K. Buchin, M. Buchin, M. van Kreveld, and J. Luo. Finding long and similar parts of trajectories. *Computational Geometry: Theory and Applications*, 44(9):465–476, 2011.

[2] Z. Chen, H. Shen, and X. Zhou. Discovering popular routes from trajectories. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*, pages 900–911. IEEE, 2011.

[3] T. Cormen. *Introduction to algorithms*. The MIT press, 2001.

[4] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 330–339. ACM, 2007.

[5] D. Guo, S. Liu, and H. Jin. A graph-based approach to vehicle trajectory analysis. *Journal of Location Based Services*, 4(3-4):183–199, 2010.

[6] L. Kaufman, P. Rousseeuw, et al. *Finding groups in data: an introduction to cluster analysis*, volume 39. Wiley Online Library, 1990.

[7] A. Kharrat, I. Popa, K. Zeitouni, and S. Faiz. Clustering algorithm for network constraint trajectories. *Headway in Spatial Data Handling*, pages 631–647, 2008.

[8] A. Lee, Y. Chen, and W. Ip. Mining frequent trajectory patterns in spatial-temporal databases. *Information Sciences*, 179(13):2218–2231, 2009.

[9] J. Lee, J. Han, and K. Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 593–604. ACM, 2007.

[10] X. Li, J. Han, J. Lee, and H. Gonzalez. Traffic density-based discovery of hot routes in road networks. *Advances in Spatial and Temporal Databases*, pages 441–459, 2007.

[11] J. Marcotty. Federal Funding for Bike Routes Pays Off in Twin Cities. `http://www.startribune.com/local/minneapolis/150105625.html`.

[12] G. Roh and S. Hwang. Nncluster: An efficient clustering algorithm for road network trajectories. In *Database Systems for Advanced Applications*, pages 47–61. Springer, 2010.

[13] D. Sacharidis, K. Patroumpas, M. Terrovitis, V. Kantere, M. Potamias, K. Mouratidis, and T. Sellis. On-line discovery of hot motion paths. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, pages 392–403. ACM, 2008.

[14] J. Won, S. Kim, J. Baek, and J. Lee. Trajectory clustering in road network environment. In *Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on*, pages 299–305. IEEE, 2009.

[15] Y. Zheng and X. Zhou. *Computing with spatial trajectories*. Springer-Verlag New York Inc, 2011.