# Fast and Exact Network Trajectory Similarity Computation: A Case-Study on Bicycle Corridor Planning

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, Environment, and Society
University of Minnesota
Minneapolis, MN, USA
fharvey@umn.edu

## ABSTRACT

Given a set of trajectories on a road network, the goal of the All-Pair Network Trajectory Similarity (APNTS) problem is to calculate the similarity between all trajectories using the Network Hausdorff Distance. This problem is important for a variety of societal applications, such as facilitating greener travel via bicycle corridor identification. The APNTS problem is challenging due to the high cost of computing the exact Network Hausdorff Distance between trajectories in spatial big datasets. Previous work on the APNTS problem takes over 16 hours of computation time on a real-world dataset of bicycle GPS trajectories in Minneapolis, MN. In contrast, this paper focuses on a scalable method for the APNTS problem using the idea of row-wise computation, resulting in a computation time of less than 6 minutes on the same datasets. We provide a case study for transportation services using a data-driven approach to identify primary bicycle corridors for public transportation by leveraging emerging GPS trajectory datasets.

## Categories and Subject Descriptors

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

## General Terms

Algorithms, Performance, Experimentation

## Keywords

Trajectory Similarity, Spatial Data Mining, Network Hausdorff Distance

## 1. INTRODUCTION

**Problem:** Given a set of trajectories on a road network, the goal of the All-Pair Network Trajectory Similarity (APNTS) problem is to calculate the Network Hausdorff Distance (NHD) between all pairs of input trajectories. The


*UrbComp'13* August 11–14, 2013, Chicago, Illinois, USA.

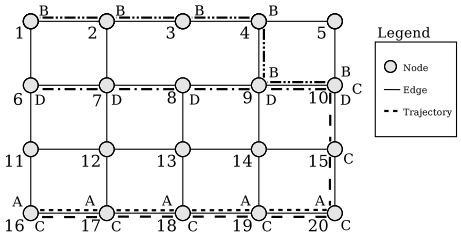




**Figure 1: Road network represented as an undirected graph with four trajectories illustrated with bold dashed lines.**

classical Hausdorff distance is defined as the "maximum distance of a set (of points) to the nearest point in the other set [17]." For example, the Network Hausdorff Distance from Trajectory B to Trajectory A in Figure 1 is 4 units (edge traversals), as every node in Trajectory B is at least 4 units or less away from any node in Trajectory A. Note that the Hausdorff distance is not symmetric. It is a commonly used similarity measure in computer vision [11] and computational geometry [10], but recently has been used to define similarity between trajectories, both in Euclidean space [15] and recently on graphs [16]. As the Hausdorff distance is set-based, it has no notion of order and therefore when applied to trajectories, traditionally time is ignored. Depending on the application domain, this may be acceptable as it is the underlying route choices that are of interest in our case study on urban bicycle corridor planning.

**Motivation:** Trajectory similarity measures are used in a number of important societal applications, such as summarizing population movement within a city, or to optimize bus route placement. Using network-based trajectories enforces topological constraints that are inherently present in road networks. Let us consider the problem of determining primary bicycle corridors through a city to facilitate safe and efficient bicycle travel. GPS trajectory data shows us where in the city bicycle commuters travel, allowing for data-driven decisions of bike corridor placement. By selecting representative corridors for a given group of commuters, we can minimize the overall alteration to their routes and encourage use of the bicycle corridors. Facilitating commuter bicycle traffic has been shown in the past to have numerous societal benefits, such as reduced greenhouse gas emissions and healthcare costs [14].

**Challenges:** The APNTS problem is challenging due to the computational cost of computing the Network Hausdorff Distance (NHD) between all pairs of input trajectories, as

it requires a large amount of node-to-node distance comparisons. In our previous work [5], we proposed an exact but expensive baseline algorithm to compute the NHD, requiring multiple invocations of common shortest-path algorithms (e.g., Dijkstra's [4]). For example, given two trajectories consisting of 100 nodes each, a baseline approach to calculate NHD would need to compare the distances for all pairs of nodes within those two trajectories ($10^4$), which over a large trajectory dataset (e.g., pair-wise comparison of 10,000 trajectories) would require $10^{12}$ distance comparisons. This quickly becomes computationally prohibitive without faster algorithms.

**Related Work:** Trajectory pattern mining is a popular field with a number of interesting problems both in geometric (Euclidean) spaces [22] and networks (graphs) [7]. A key component to traditional data mining in these domains is the notion of a similarity metric, the measure of sameness or closeness between a pair of objects. A variety of trajectory similarity metrics, both geometric and network, have been proposed in the literature. One popular metric is Hausdorff distance, a commonly used measure to compare similarity between two geometric objects (e.g., polygons, lines, sets of points) [11]. A number of methods have focused on applying Hausdorff distance to trajectories in geometric space [2, 3, 9, 15].

Hausdorff distance has been shown to be a useful tool in geometric space for measuring similarity between trajectories for applications that do not use the temporal information of trajectories, but applying Hausdorff distance to network-based trajectories is non-trivial. A number of papers have proposed heuristics to approximate the Hausdorff distance on networks [12, 13, 16, 18, 19]. This is due to the large number of graph-distance computations needed to compute the NHD. These approximations allow for interesting and useful pattern discovery, but do not compute exact similarities between trajectories and may alter results. In our previous work [5], we proposed a correct but computationally expensive algorithm for clustering network trajectories on road networks using Network Hausdorff Distance to identify new bicycle corridors through a city to facilitate safe and efficient bicycle travel. However, while the optimized algorithm was a significant improvement over the baseline, it still remained computationally prohibitive for large trajectory datasets. We illustrate a classification of related work in Figure 2, highlighting the various approaches of Hausdorff distance computation for trajectories. In this paper, we propose a new algorithm that improves performance by over an order of magnitude on synthetic and case study datasets.

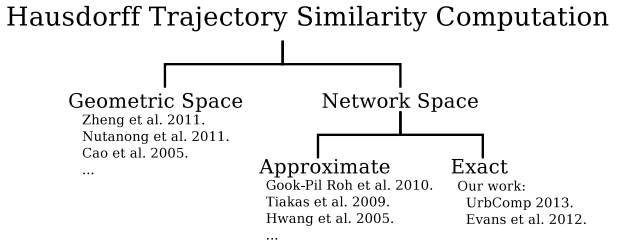



**Figure 2: Classifications of Hausdorff Trajectory Similarity Algorithms.**

**Proposed Approach:** In this paper, we formalize the Network Hausdorff Distance for weighted graphs and propose a novel approach that is orders of magnitude faster than the baseline approach and our previous work, allowing for Network Hausdorff Distance to be computed efficiently on large trajectory datasets. We propose a novel approach that computes the exact NHD while remaining computationally efficient. While the baseline approach computes node-to-node distances, the Network Hausdorff Distance (NHD) essentially requires node-to-trajectory minimum distance values. We take advantage of this insight to compute the NHD between nodes and trajectories directly by modifying the underlying graph, inserting a super-source node [6] and computing from a trajectory to an entire set of trajectories with a single shortest-paths distance computation.

**Contributions:** This paper proposes a number of key contributions:

- We formalize the All-Pair Network Trajectory Similarity (APNTS) problem for computing similarities between a set of trajectories.
- We propose a fast and exact algorithm, ROW-TS, to solve the APNTS problem.
- We provide a case study on real-world GPS trajectory data of bicycle commuters in Minneapolis, MN.
- We validate the ROW-TS algorithm with experimental analysis.

**Scope** This paper focuses on network trajectory similarity computation, being a crucial step for a number of trajectory pattern mining algorithms (e.g., trajectory clustering algorithms), as illustrated in our previous work which focuses on clustering for primary corridors in transportation [5]. A key component of the Network Hausdorff Distance is shortest-path distance computations and values. In this paper, we will discuss a Dijkstra-like [4] framework for computing these paths during trajectory similarity computation. This assumes that the underlying network is too large to precompute and store the all-pair shortest-path distance matrix (e.g., Floyd-Warshall [4]). We ignore order-dependent trajectories (chronological ordering, etc.) for simplicity, but the discussed techniques can be transferred to a spatio-temporal network if a relation between space and time is needed.

**Outline:** The rest of the paper is organized as follows. A brief description of the basic concepts and problem formulation is presented in Section 2. A description of the computational structure of the APNTS problem is presented in Section 3. In Section 4 we propose the ROW-TS algorithm. We provide a case-study on a real-world bicycle GPS trajectory dataset in Section 5 and experimental analysis on synthetic datasets in Section 6. Finally, we conclude in Section 7 with a discussion of future work.

# 2. PROBLEM FORMULATION

In this section, we describe the basic concepts required to describe the All-Pair Network Trajectory Similarity (APNTS) problem. We provide an example dataset and a formal problem statement.

## 2.1 Basic Concepts

We begin with a review of trajectories, networks, and how they are used to calculate the Hausdorff distance.

*Definition 1.* **Road Network**

A road network is defined in this paper as an undirected, weighted graph $G = \{V, E\}$ illustrated in Figure 1 where $V = (1, 2, 3, ..., 18, 19, 20)$ are the vertices or nodes in the graph $G$ and $E = (1-2, 1-6, ..., 18-19, 14-19, 19-20...)$ are the edges connecting the vertices (edge weights of 1 unit). This representation is commonly used for transportation networks, where intersections are modeled as nodes. Note that the graph could be directed without a change to the proposed approach.

*Definition 2.* **Network Trajectory**

A spatial trajectory traditionally refers to a series of points (a trace) of a moving object in geographic space [22]. In this paper, we will be focusing on network trajectories, or, trajectories that consist as a set of nodes and edges in a graph. A common operation for GPS trajectory data on road networks involves map-matching the GPS points to the network [22]. This allows describing the trajectory using graph notation (a set of nodes and edges) and perform graph-based calculations, such as shortest-path distance. In this paper, we define network trajectories as a set of connected vertices: $t_x = [v_1, ..., v_n]$. Figure 1 contains four network trajectories: $t_A$ = [16, 17, 18, 19, 20]; $t_B$ = [1, 2, 3, 4, 9, 10]; $t_C$ = [16, 17, 18, 19, 20, 15, 10]; $t_D$ = [6, 7, 8, 9, 10].

*Definition 3.* **Shortest-Path Distance**

To measure shortest-path distance on a network or graph, we sum of the length of edges required to traverse the graph between two specific nodes. A number of well-known algorithms compute shortest-paths (shortest path tree) on graphs [4]. In this paper, we will use the generic function $dist(n, m)$ to indicate the shortest-path distance between nodes $n$ and $m$. The choice of algorithm used to calculate this value may vary (Dijkstra's, A*, Floyd-Warshall if network size is small). In our pseudocode and implementation, we use Dijkstra's single-source shortest-paths algorithm [4] taking a source node $v$ and a graph $G$ as input: $distMap[] = Dijkstra(G, v)$. It returns the shortest-path distance to all other nodes in $G$. In Figure 1, an example $dist(3, 8)$ would be a distance of 3, traversing the shortest-path 3-2-7-8 or 3-4-9-8.

*Definition 4.* **Network Hausdorff Distance**

As discussed in the related work, there are a number of variations and approximations of Hausdorff distance on networks [12, 13, 16, 18, 19]. Based on these and the original Hausdorff distance in geometric space [1], we formulate Network Hausdorff Distance (NHD) between two trajectories $t_x$ and $t_y$ in Equation 1. An example of NHD, and its asymmetric distances is $NHD(t_A, t_B)$ and $NHD(t_B, t_A)$. $NHD(t_A, t_B) = 3$ as the farthest node in $t_A$ from $t_B$ is 3 edges away and tied between a number of pairs: $(dist(16_A, 1_B) = 3)$, $(dist(17_A, 2_B) = 3)$, $(dist(18_A, 9_B) = 3)$. $NHD(t_B, t_A) = 4$ as the farthest node in $t_B$ from $t_A$ is 4 edges away with: $(dist(3_B, 17_A) = 4$ and $dist(3_B, 19_C) = 4$.

$$NHD(t_i, t_j) = \max_{n \in t_i} \min_{m \in t_j} dist(n_{t_i}, m_{t_j}) \quad (1)$$

*Definition 5.* **Trajectory Similarity Matrix**

**Table 1: Output for the All-Pair Network Trajectory Similarity problem: a Trajectory Similarity Matrix for the input data in Figure 1 using Network Hausdorff Distance.**

| $NHD(t_x, t_y)$ | Track A | Track B | Track C | Track D |
|---|---|---|---|---|
| Track A | 0 | 3 | 0 | 2 |
| Track B | **4** | 0 | 3 | 2 |
| Track C | 2 | 3 | 0 | 2 |
| Track D | 2 | 1 | 2 | 0 |

A similarity matrix contains the values based on some measure comparing each pairwise combination of objects in a dataset. They are input for a number of data mining problems (e.g., clustering, classification). In this paper, our focus is to quickly and efficiently produce a similarity matrix of input trajectories for use in popular trajectory pattern mining algorithms [22]. Table 1 contains the trajectory similarity matrix for the APNTS problem on the input data in Figure 1. Each cell value is the result of the $NHD(t_x, t_y)$ equation given the two corresponding tracks (row, column).

## 2.2 Problem Statement

The All-Pair Network Trajectory Similarity (APNTS) problem can be formulated as follows:

**Input:**
- Road Network: $G = \{V, E\}$
- Collection of Trajectories: $T$

**Output:**
- Trajectory Similarity Matrix: $M : T \times T \rightarrow \mathbb{R}$

**Objective:**
- Minimize computation time

**Constraints:**
- $M$ values are correct given Equation 1 (Network Hausdorff Distance)
- $G$ is a undirected, weighted graph with nonnegative edge weights
- Trajectories in $T$ are paths in $G$

**Example:** Given the input road network and four trajectories in Figure 1, the corresponding trajectory similarity matrix is shown in Table 1. For each cell in the matrix, the NHD value was calculated based on the two input trajectories.

# 3. COMPUTATIONAL STRUCTURE

Network Hausdorff Distance is a recently popular topic in the literature [12, 13, 16, 18–20]. However, each of these papers cite performance issues with a network-based Hausdorff distance computation, with [16] saying "the baseline algorithm requires a large number of distance computations which significantly degrades performance...". This idea is echoed in the other papers, each proposing interesting and novel algorithms to approximate a network-based Hausdorff distance computation. In this section, we will discuss the underlying computational structure of this baseline algorithm and mention a few of the approximation algorithms and their limitations.

Essentially, the related work provides algorithms to reduce the number of Network Hausdorff Distance computations between trajectories using clustering techniques (e.g.,

density-based [20] or $k$-nn [16]) or by converting network trajectories to geometric space [21]. These clustering approaches degrade to the baseline brute-force approach when threshold values (density and neighborhood) are not appropriately set. In addition, as some portion (depending on thresholds) of trajectories are not being compared via the Network Hausdorff Distance, the resulting distance values are not exact and may affect the quality of the results of any algorithm using such values. In this paper, we focus on an exact approach, ensuring each trajectory pair similarity is computed with the exact Network Hausdorff Distance while retaining computational scalability.

**Table 2: Network distance between node pairs; required for $NHD(t_B, t_A)$ (Input: Figure 1, Full trajectory similarity matrix shown in Table 1.**

| $NHD(t_B, t_A)$ | $16_{t_A}$ | $17_{t_A}$ | $18_{t_A}$ | $19_{t_A}$ | $20_{t_A}$ | Min |
|---|---|---|---|---|---|---|
| $1_{t_B}$ | 3 | 4 | 5 | 6 | 7 | 3 |
| $2_{t_B}$ | 4 | 3 | 4 | 5 | 7 | 3 |
| $3_{t_B}$ | 5 | 4 | 5 | 4 | 5 | 4 |
| $4_{t_B}$ | 6 | 5 | 4 | 3 | 4 | 3 |
| $9_{t_B}$ | 5 | 4 | 3 | 2 | 3 | 2 |
| $10_{t_B}$ | 6 | 5 | 4 | 3 | 2 | 2 |
| Max | - | - | - | - | - | **4** |

## 3.1 Graph-Node Track Similarity Baseline (GNTS - B)

The baseline algorithm to compute the Network Hausdorff Distance Track Similarity Matrix $M$ computes the shortest-path distance between each pair of nodes within each pair of trajectories, choosing the maximum of the values of this set. Due to this enumeration, the approach is given the name Graph-Node Track Similarity, as it focuses on repeated graph computations between all pairs of nodes within the two trajectories being compared. For example, in Figure 1, to compute the similarity between Trajectory $t_B$ and Trajectory $t_A$, we need to find the minimum distance from each node in Trajectory $t_B = [1_{t_B}, 2_{t_B}, 3_{t_B}, 4_{t_B}, 9_{t_B}, 10_{t_B}]$ to any node in Trajectory $t_A = [16_{t_A}, 17_{t_A}, 18_{t_A}, 19_{t_A}, 20_{t_A}]$, as described in Equation 1. These necessary shortest-path distances are shown in Table 2. These values can be computed via any number of shortest-path algorithms, but for simplicity we will focus on the popular Dijkstra's single-source shortest-paths algorithm [4]. Computing Table 2 would require 6 (or $|t_B|$) invocations of Dijkstra's algorithm to find the shortest-path distance from each node in $t_B$ to each node in $t_A$ with a runtime of $O(|V|log|V| + |E|)$ [4]. That amount of computation is necessary to find $NHD(t_B, t_A)$, one cell in the trajectory similarity matrix (TSM) shown in Table 1. Therefore, in the baseline case, a number of shortest-path algorithm invocations are required for each cell in the TSM.

# 4. PROPOSED APPROACH

In this section we will describe our proposed approach to efficiently solve the APNTS problem. We begin by proposing our novel row-wise trajectory similarity (ROW-TS) algorithm. The ROW-TS algorithm calculates an entire row of the trajectory similarity matrix described in the APNTS problem with a single shortest-path computation.

## 4.1 Row-Wise Track Similarity (ROW-TS)

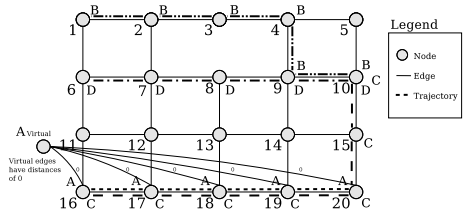



**Figure 3: Inserting a virtual node ($A_{virtual}$) to represent Track A for efficient Network Hausdorff Distance computation.**

Computing the Network Hausdorff Distance $NHD(t_x, t_y)$ between two trajectories does not require the shortest distance between all-pairs of nodes in $t_x$ and $t_y$. Instead, it requires the minimum distance from each node in $t_x$ to the *closest* node in $t_y$. In Figure 3, to calculate $NHD(t_B, t_A)$, we begin by inserting a virtual node ($A_{virtual}$) representing Trajectory $t_A$ into the graph. This node has edges with weights of 0 connecting it to each other node in Trajectory $t_A$. We then run a shortest-path distance computation from the virtual node as a source, with the destination being every node in Trajectory $t_B$. The result was the shortest distance from each node in Trajectory $t_B$ to the virtual node $A_{virtual}$. Since the virtual node is only connected to nodes in Trajectory $t_A$, and all the edge weights are 0, we had the shortest-path from each node in Trajectory $t_B$ to the closest node in Trajectory $t_A$, exactly what $NHD(t_B, t_A)$ requires for computation. However, this focused on computing a single cell in the trajectory similarity matrix per invocation of a single-source shortest-paths algorithm. That means at least $O(|T|^2)$ shortest-path invocations to compute the TSM for the APNTS problem, still quite expensive. We propose a new approach, ROW-TS, to compute an entire *row* of the TSM with one invocation of a single-source shortest-paths algorithm. Using a row-based approach, we can essentially calculate $NHD(t \in T, t_A)$ with one single-source shortest-paths invocation from $A_{virtual}$. This approach reduces the overall number of shortest-paths invocations to $O(|T|)$ at the cost of additional bookkeeping, as we will show below.

The pseudocode for ROW-TS is given in Algorithm 1. To compute the TSM in Table 1, ROW-TS iterates through each input trajectory $t_x$. In Lines 2 - 6, a virtual node is added to the graph and connected to each node $t_x$ by an edge of length zero. In line 7, ROW-TS runs an undirected shortest path tree algorithm (Dijkstra's [4]) to find the shortest-path distance between all tracks and the virtual node, updating a distance map. In lines 9-10, ROW-TS iterates through trajectories in $T$ and tests if the network Hausdorff Distance from one trajectory to itself is being computed, which is always zero. In lines 13-19 ROW-TS checks the distance map returned by the shortest-path computation to get the Hausdorff distance. In line 19, ROW-TS updates the track similarity matrix $M$ with the newly calculated Network Hausdorff Distance between $t_x$ and $t_y$ and returns $M$ in line 23.

**Execution Trace of ROW-TS:** We begin by iterating through each trajectory $t_x$ in $T$. Let's use Trajectory $t_A$ in Figure 1 as an example trajectory through this algorithm. In Line 2 we create a virtual node to represent Trajectory $t_A$ ($A_{virtual}$ in Figure 3). Connecting $A_{virtual}$ to each node in Trajectory $t_A$ (16, 17, 18, 19, 20) with 0-length edges,

**Algorithm 1** Row-Wise Track Similarity (ROW-TS)

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

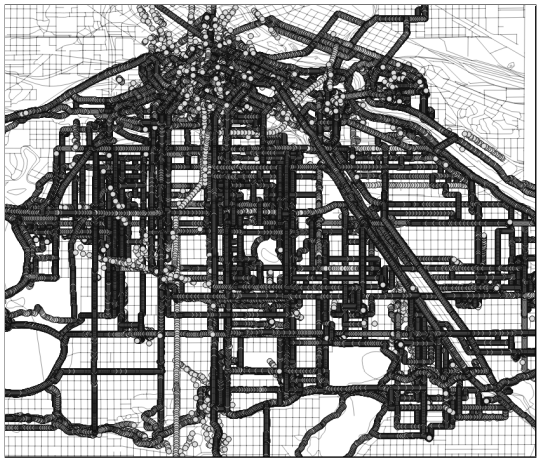
**Output:**
- Track Similarity Matrix $M$

```
1: for t_x in T do
2:    vTrack = newNode()
3:    G.add(vTrack)
4:    for nodes n in t_x do
5:       Create 0-length edge from vTrack to n
6:    end for
7:    distMap[] = Dijkstra(G, vTrack)
8:    G.remove(vTrack)
9:    for t_y in T do
10:      if t_x == t_y then
11:         M[t_y][t_x] = 0
12:      else
13:         max = −∞
14:         for nodes m in t_y do
15:            if distMap[m] ≥ max then
16:               max = distMap[m]
17:            end if
18:         end for
19:         M[t_y][t_x] = max
20:      end if
21:   end for
22: end for
23: return M
```

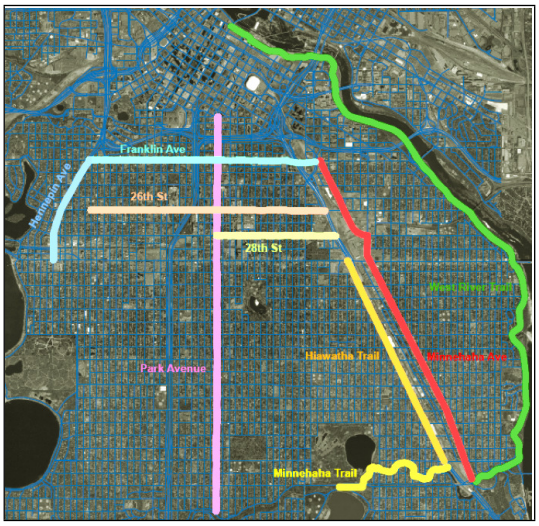
we can compute the single-source shortest-paths distance from $A_{virtual}$ to every other node in the graph $G$ in Line 7. Since the edges connected to $A_{virtual}$ had a length (weight) of 0, we know that each node $n$ distance returned in the $distMap[]$ on Line 7 is actually the shortest-path distance from $n$ to some node in $A$. This set of minimum distances is what we need to calculate the NHD from each trajectory $t$ in $T$, which we loop through in Line 9. For each node $m$ in trajectory $t$, we already have the minimum distance from $m$ to any node in trajectory $A$ in the $distMap[]$, and therefore need to find the maximum of those minimums as defined in Definition 4 for the NHD($t$,$t_A$). For an example, lets use Trajectory $t_B$ as $t$ starting in Line 9. We iterate through each node $m_B$ in $t_B$ ($1_{t_B}, 2_{t_B}, 3_{t_B}, 4_{t_B}, 9_{t_B}, 10_{t_B}$) and lookup the distance from $A_{virtual}$ to $m_B$ in $distMap[]$ ($1_{t_B}$:3, $2_{t_B}$:3, $3_{t_B}$:4, $4_{t_B}$:3, $9_{t_B}$:2, $10_{t_B}$:2) as shown in Table 2. The maximum of those minimum distances is 4, which we set as the value of $NHD(t_B, t_A)$ in Line 19. This repeats for every other trajectory $t$ in $T$ to compute the entire row corresponding to Trajectory $t_A$ in the trajectory similarity matrix $M$.

## 5. CASE STUDY: K-PRIMARY CORRIDORS FOR COMMUTER BICYCLISTS

In Summer 2006, University of Minnesota researchers collected a variety of data to help gain a better understanding of commuter bicyclist behavior using GPS equipment and personal surveys to record bicyclist movements and behaviors [8]. One possible issue they looked at was identifying popular transportation corridors for the Minnesota Department of Transportation to focus funds and repairs. At the



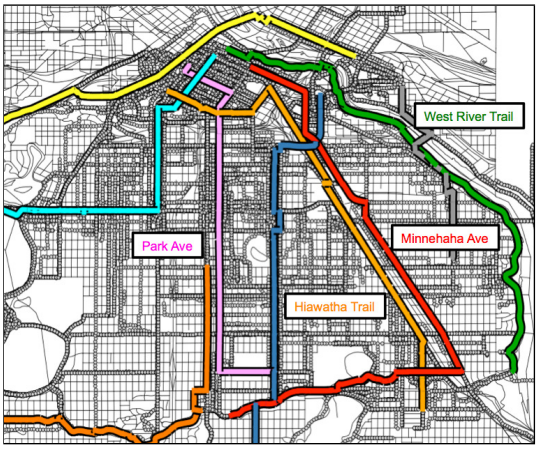
(a) Recorded GPS points from bicyclists in Minneapolis, MN. Intensity indicates number of points.



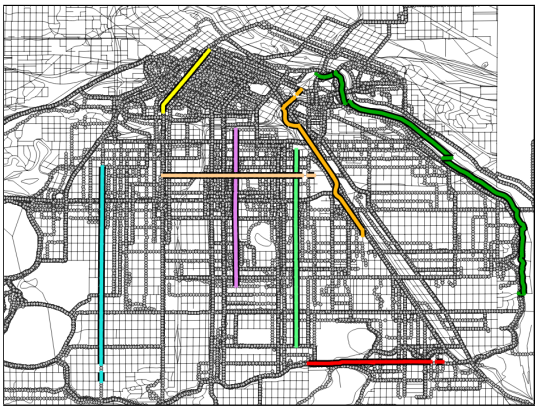

(b) Original, hand-crafted primary corridors identified by our co-author and fellow geographers in [8]

**Figure 4: Example input and output of the $k$-Primary Corridor problem.**

time, they hand-crafted the primary corridors. Shortly after this study, the U.S. Department of Transportation began a four-year, $100 million pilot project in four communities (including Minneapolis) aimed to determine whether investing in bike and pedestrian infrastructure encouraged significant increases in public use. As a direct result of this project, the US DoT found that biking increased 50%, 7,700 fewer tons of carbon dioxide were emitted, 1.2 million fewer gallons of gas was burned, and there was a $6.9 million/year reduction in health care costs [14]. In our previous work [5], we proposed a data-driven approach (the METS algorithm) to identify these primary bike corridors. While this approach identified interesting corridors, it remained computationally expensive as shown in Table 3. We re-ran the case-study in [5] on the Minneapolis, MN bicycle GPS dataset using the METS and ROW-TS algorithms in Table 3, given the input of the Hennepin County road network (57,644 nodes), 819 trajectories (shown in Figure 4(a) and the number of desired corridors, $k$, set to 8. As expected, the ROW-TS algorithm significantly outperformed the METS algorithm. The faster




(a) 8-Primary Corridors chosen from input GPS.



(b) 8-Primary Corridors chosen from subset of single streets (e.g., Park Ave)

**Figure 5: Set of *8*-primary corridors chosen based on minimized intra-cluster distance. The set of candidate corridors varies in each figure.**

execution allowed a modification to the case-study to help identify more appropriate corridors.

**Table 3: CPU Execution Time on Bicycle GPS trajectories in Minneapolis, MN**

| Dataset | METS | ROW-TS |
|---|---|---|
| 57,644 Nodes, 819 Tracks | 55,823 sec | 353.9 sec |

In the previous case-study, the chosen primary corridors where selected from the input trajectories (a k-medoid approach). While this ensured that chosen corridors were routes that had been biked in the real world and shared commonalities with the hand-crafted corridors chosen in Figure 4(b), they had problems of being too long or having too many turns and deviations. In Figure 5(a), we show the corridors chosen from the input trajectories. For this case-study, we generated a set of candidate corridors based on input from geographers and had the ROW-TS algorithm compute the similarity between these candidates and the input GPS trajectory data. The candidate corridors were generated with two restrictions: 1) the corridor was a shortest-path between two nodes on the network, and 2) the corridor could only consist of $n$ different streets, defined by their street name (e.g., a corridor traversing Park Ave and Cedar Ave would consist of 2 streets). This approximated the types of corridors our geographer collaborators were looking for while retaining the data-driven approach by comparing them to the actual GPS trajectory data. For this study, we generated 5,000 trajectories using these two constraints (source and destination nodes chosen at random) to be used as candidate corridors. The faster ROW-TS algorithm allowed us to generate a large number of candidate corridors and compute the similarity for the clustering. In Figure 5(b), we show the chosen clusters generated from single-road candidates (e.g., a subset of Park Ave would be a candidate).

# 6. EXPERIMENTAL EVALUATION

In this section, we explain our experimental design for investigating a number of computational questions related to the APNTS problem and our proposed solution. We begin with the overall experimental design, followed by an explanation of the explored parameters, and finally the experimental validation.

## 6.1 Experimental Goals

We explored a number of computational questions related to the scalability of the algorithms we proposed. Our intention was to examine how varying certain key parameters would affect the runtime of the GNTS, METS, and ROW-TS algorithms. Each question was run on a small dataset and a large dataset for each algorithm. The questions we explored were:

- How does the number of *trajectories* affect the computational cost of each algorithm?
- What is the impact of the *number of nodes* in the road network on the computational cost of the various trajectory similarity algorithms?
- Does the *trajectory length* significantly affect the runtime of each algorithm?

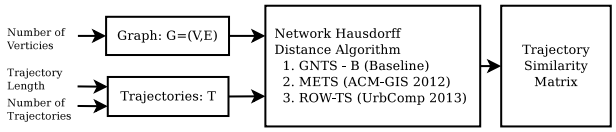



**Figure 6: Experiment Design**

## 6.2 Experimental Design

The experiments in this section were performed on synthetic datasets. We generated the underlying roadmap (graph), along with suitable trajectories. To generate the roadmap, we created a generator that required a number of nodes as input, and then generated a grid-like planar graph, similar to urban road networks. For the trajectories, we chose two random nodes on the graph and found the shortest-path, using that as a trajectory. We did not attempt to simulate how bikers would move about our networks, as we have real data (in the previous section) to test on. Here we are simply testing for scalability. These experiments were carried out on a Quad-core 2.4 Ghz Intel Core Duo 2 Ubuntu Linux machine with 8 GB of RAM. All of the algorithms were coded and run with Java SE 1.6. Each test measured CPU execution time and was run 10 times for each parameter value, then averaged for a final score shown in the plots. Our intention with these experiments was to

vary key input parameters to measure how they affect the performance of our proposed work ROW-TS using the experiment design in Figure 6. We will be comparing against related work with the following algorithms:

**Graph-Node Track Similarity (GNTS - Baseline):** The baseline algorithm to compute the Network Hausdorff Distance Track Similarity Matrix $M$ computes the shortest-path distance between each pair of nodes within each pair of trajectories, choosing the maximum of the values of this set.

**Matrix-Element Track Similarity (METS [5]):** In our previous work [5], we proposed a correct but computationally expensive algorithm for clustering network trajectories on road networks using Network Hausdorff Distance for identifying new bicycle corridors through a city to facilitate safe and efficient bicycle travel. This algorithm used a virtual node to reduce the number of shortest-path invocations to one per cell in the trajectory similarity matrix. It proved to be faster than the baseline, but remained computationally prohibitive for large datasets.

## 6.3 Experimental Results

**How does the number of *trajectories* affect the computational cost of each algorithm?** In the first experiment, shown in Figure 7(a), we created a road network with 500 nodes, an average trajectory length of 50 nodes. The x-axis shows the number of trajectories 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 trajectories, and as is clear, GNTS and METS quickly become prohibitive even with this small dataset size. ROW-TS grows slowly and appears to run at least two orders of magnitude faster. In Figure 7(d) we increased the number of trajectories to be tested, withdrawing GNTS due to prohibitive computation time. METS was also computationally prohibitive after an input of 4,000 trajectories, while ROW-TS continued to be reasonable (under 100 seconds) past 10,000 input trajectories.

**Does the *trajectory length* significantly affect the runtime of each algorithm?** We ran this experiment with 500 nodes and 500 trajectories at a trajectory length shown on the x axis in Figure 7(c). METS and ROW-TS are not significantly affected by the smaller dataset. Moving to the larger dataset in Figure 7(f) of 1,000 nodes and 500 tracks with longer trajectory lengths on the x axis, METS is not significantly affected, as expected because the main cost is still the repeated shortest-path computations. ROW-TS grows more quickly due to the extra bookkeeping in the algorithm (see the pseudocode in Algorithm 1).

**What is the impact of the *number of nodes* in the road network on the computational cost of the various trajectory similarity algorithms?** In Figure 7(b), it is clear that the size of the underlying graph (number of nodes as plotted on the x axis) is relevant to the runtime of each algorithm as shortest-path algorithms make up a large portion of the computational cost in each. We ran this experiment with 100 trajectories and an average trajectory length of 10. Figure 7(e) shows a larger dataset with the number of nodes in the network going up to 10,000. Due to the reduced number of distance calculations in ROW-TS, not only is the overall runtime differing by orders of magnitude, the growth of ROW-TS compared to GNTS is also significantly slower.

# 7. CONCLUSION

In this paper, we formalized the All-Pair Network Trajectory Similarity (APNTS) problem. We proposed a novel algorithm (ROW-TS) for quickly solving this problem and compared it to previous and related work (METS, GNTS). Calculating similarity between network trajectories has shown to be important in a number of societal applications, such as bike corridor planning and transportation management. Related work in the field have repeatedly used the Network Hausdorff Distance (NHD) measure to compute similarities, but have proposed approximation-based approaches to reduce computation time. Our previous work introduced an algorithm to compute NHD exactly and scaled to a medium-sized dataset. Our novel row-wise approach in this paper allowed us to scale to spatial big data and compute the NHD between tens of thousands of trajectories in a relatively short time. We demonstrated the scalability of ROW-TS with experimental validation.

In the future, we plan to explore more applied use of the APNTS solutions provided by this paper. While so far we have focused on computing a fully-materialized trajectory similarity matrix, we will explore computing Network Hausdorff Similarity between trajectories on the fly as clustering algorithms are asking for similarities. This will reduce memory requirements allowing for trajectory clustering on larger datasets.

# 8. 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. We would like to thank Kim Koffolt and the members of the Spatial Databases and Spatial Data Mining research group at the University of Minnesota for their helpful comments.

# 9. REFERENCES

[1] Helmut Alt and Leonidas J Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. *Handbook of computational geometry*, 1:121–153, 1999.

[2] Hu Cao and Ouri Wolfson. Nonmaterialized motion information in transport networks. *Database Theory-ICDT 2005*, pages 173–188, 2005.

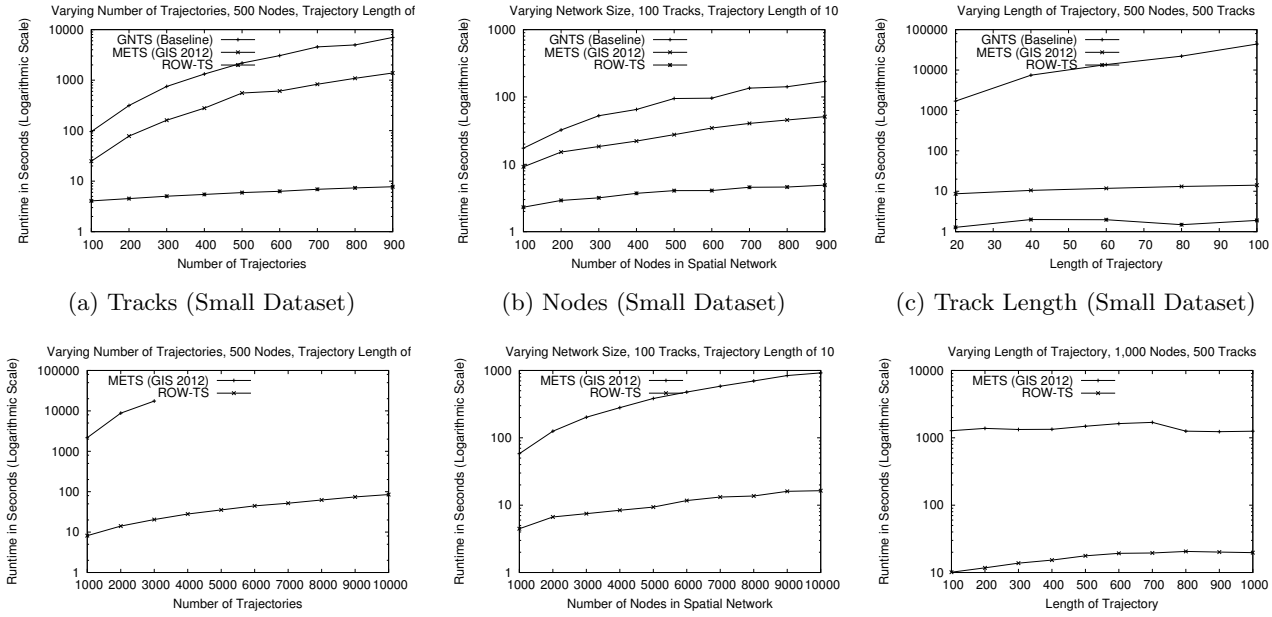
[3] Jinyang Chen, Rangding Wang, Liangxu Liu, and Jiatao Song. Clustering of trajectories based on hausdorff distance. In *Electronics, Communications and Control (ICECC), 2011 International Conference on*, pages 1940–1944. IEEE, 2011.

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

[5] Michael R. Evans, Dev Oliver, Shashi Shekhar, and Francis Harvey. Summarizing trajectories into k-primary corridors: a summary of results. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 454–457, New York, NY, USA, 2012. ACM.

[6] D. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, USA, 2010.

[7] Ralf Hartmut Güting, Victor Teixeira De Almeida, and Zhiming Ding. Modeling and querying moving




(a) Tracks (Small Dataset) (b) Nodes (Small Dataset) (c) Track Length (Small Dataset)

(d) Tracks (Large Dataset) (e) Nodes (Large Dataset) (f) Track Length (Large Dataset)

**Figure 7: Experimental results on synthetic data. Note the y-axis is in logarithmic scale.**

objects in networks. *The VLDB Journal*, 15(2):165–190, 2006.

[8] F.J. Harvey and K.J. Krizek. Commuter Bicyclist Behavior and Facility Disruption. Technical Report Report no. MnDOT 2007-15, University of Minnesota, 2007.

[9] Jeff Henrikson. Completeness and total boundedness of the hausdorff metric. *MIT Undergraduate Journal of Mathematics*, 1:69–79, 1999.

[10] Daniel P Huttenlocher and Klara Kedem. Computing the minimum hausdorff distance for point sets under translation. In *Proceedings of the sixth annual symposium on Computational geometry*, pages 340–349. ACM, 1990.

[11] Daniel P Huttenlocher, Klara Kedem, and Jon M Kleinberg. On dynamic voronoi diagrams and the minimum hausdorff distance for point sets under euclidean motion in the plane. In *Proceedings of the eighth annual symposium on Computational geometry*, pages 110–119. ACM, 1992.

[12] Jung-Rae Hwang, Hye-Young Kang, and Ki-Joune Li. Spatio-temporal similarity analysis between trajectories on road networks. In *Proceedings of the 24th international conference on Perspectives in Conceptual Modeling*, ER'05, pages 280–289, Berlin, Heidelberg, 2005. Springer-Verlag.

[13] Jung-Rae Hwang, Hye-Young Kang, and Ki-Joune Li. Searching for similar trajectories on road networks using spatio-temporal similarity. In *Advances in Databases and Information Systems*, pages 282–295. Springer, 2006.

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

[15] Sarana Nutanong, Edwin H Jacox, and Hanan Samet. An incremental hausdorff distance calculation algorithm. *Proceedings of the VLDB Endowment*, 4(8):506–517, 2011.

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

[17] Günter Rote. Computing the minimum hausdorff distance between two point sets on a line under translation. *Information Processing Letters*, 38(3):123–127, 1991.

[18] E. Tiakas, A. N. Papadopoulos, A. Nanopoulos, Y. Manolopoulos, Dragan Stojanovic, and Slobodanka Djordjevic-Kajan. Searching for similar trajectories in spatial networks. *J. Syst. Softw.*, 82(5):772–788, May 2009.

[19] Eleftherios Tiakas, Apostolos N Papadopoulos, Alexandros Nanopoulos, Yannis Manolopoulos, Dragan Stojanovic, and Slobodanka Djordjevic-Kajan. Trajectory similarity search in spatial networks. In *IDEAS'06. 10th International*, pages 185–192. IEEE, 2006.

[20] Yalin Wang, Qilong Han, and Haiwei Pan. A clustering scheme for trajectories in road networks. In *Advanced Technology in Teaching-Proceedings of the 2009 3rd International Conference on Teaching and Computational Science (WTCS 2009)*, pages 11–18. Springer, 2012.

[21] Hongbin Zhao, Qilong Han, Haiwei Pan, and Guisheng Yin. Spatio-temporal similarity measure for trajectories on road networks. In *Internet Computing for Science and Engineering, Fourth International Conference on*, pages 189–193. IEEE, 2009.

[22] Yu Zheng and Xiaofang Zhou. *Computing with Spatial Trajectories*. Springer Publishing Company, Incorporated, 1st edition, 2011.