Edit distance with Quasi Real Penalties: a hybrid distance for network-constrained trajectories

Noudehouenou L.J. Houssou, Jean-Loup Guillaume, Armelle Prigent

Edit distance with Quasi Real Penalties: a hybrid distance for network-constrained trajectories

Noudéhouénou L. J. Houssou
Laboratory L3i
La Rochelle University
La Rochelle, France
ORCID 0000-0003-1070-5154

Jean-loup Guillaume
Laboratory L3i
La Rochelle University
La Rochelle, France
ORCID 0000-0002-4615-1563

Armelle Prigent
Laboratory L3i
La Rochelle University
La Rochelle, France
armelle.prigent@univ-lr.fr

***Abstract*—In this paper, we propose a new distance for network-constrained trajectories named Edit distance with Quasi Real Penalties (*EQRP*). Depending on the case, it can compare trajectories as non-ordered sets and as sequences while other distances only compare trajectories as non-ordered sets or as sequences. Moreover, it is parameter-free, manages local time shifting, and respects triangle inequality; three properties expected from a trajectory distance that are not satisfied simultaneously by any other distance to the best of our knowledge. To demonstrate the pertinence of our idea, we benchmark our distance against some state-of-the-art distances for network-constrained trajectories. Specifically, for each distance, we determine its capability to identify precisely similar trajectories. We also determine their respective performance for trajectory clustering. Our results show the predominance of *EQRP* over the existing edit distances and in some cases a more precise ability to evaluate the dissimilarity between network-constrained trajectories compared to other measures.**

***Index Terms*—Trajectory analysis, edit distance, road network, clustering, graphs.**

I. Introduction

Location-based services (LBS) are widely used in our modern societies to provide assistance in many contexts such as social interactions, navigation, entertainment, or fitness activities for example. This results in massive production and storage of spatio-temporal traces also called trajectories, which mainly consist of ordered sequences of locations and describe their authors' movements in time and space. We distinguish two kinds of trajectories: free-space or nearly free-space ones which can take any form and occur in an environment with no or few limitations (e.g. on the sea) and network-constrained trajectories corresponding to travel that only take place along the edges of a given network such as a road network (see Figure 1). Compared to free-space trajectories, network-constrained trajectories are more related to human activities and have broader socio-economic applications. They are for example essential to understand mobility patterns in cities, predict traffic congestion, select the best advertisement and business locations or recommend optimal travel routes.

Initially, network-constrained trajectories were processed without taking into account the underlying network structure as their free-space counterpart. This is because most methods and measures dedicated to trajectory analysis have been originally designed for free-space trajectories. However, such an approximation can induce significant bias in trajectories comparison. For example, trajectories can be mistakenly declared similar because some of their points appear spatially close while the network distance between them is actually large. To overcome these biases, a bunch of measures specially developed for network-constrained trajectories have been recently proposed in the literature. We can classify them into four main families: simple distances, node-to-trajectory distances, set distances, and warping distances. Simple distances compare trajectories by evaluating the distance between nodes of the same index (i.e. comparing first node of each trajectory, then second node, and so on) or using a set of points of interest. Node-to-trajectory distances determine the minimum spacing between each node of a first trajectory and all the nodes of a second one. Set distances compare two trajectories based on the size of their intersection, i.e. the number of nodes or edges they have in common. Finally, warping distances take into account the order of the nodes and try to find the optimal alignment between trajectories when comparing them. This brings an additional dimension to the trajectories comparison process which is not present in the three other families.

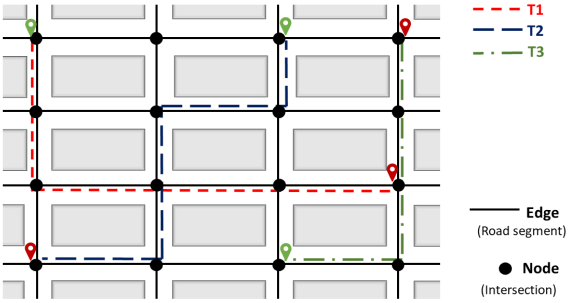

Fig. 1: Network-constrained trajectories.

While warping distances consider trajectories as sequences of nodes and the others mostly as non-ordered sets of nodes, trajectories actually bear both characteristics and should be compared along these two dimensions. For example, a trajectory and its copy in the opposite direction have exactly the same nodes but are not identical, without being completely different. However, to the best of our knowledge, none of the existing distances are able to process trajectories both as non-

ordered sets and sequences. In addition, we compared state-of-the-art distances by checking three major properties: triangle inequality, local time shifting handling, and parameter dependence. Triangle inequality ensures that the distance measure discriminates correctly the compared trajectories. It is also used to reduce the number of distance computations through pruning strategies when comparing large sets of trajectories. Local time shifting corresponds to the presence of very similar sub-sequences, offset from each other, in different trajectories. A measure able to handle it can optimally align trajectories during their comparison. Concerning parameter dependence, we are actually evaluating if the measure is parameter-free or not as it is more suitable to have no parameters to calibrate before applying a measure. None of the existing distances satisfy simultaneously these three properties even though the shortcomings of some distances are compensated by other distances and vice-versa. These observations led us to formulate the following hypothesis: is it possible to hybridize two distances to obtain a measure that respects the three forementioned properties but also treats trajectories as non-ordered sets and sequences concomitantly? We respond positively to this question by proposing Edit distance with Quasi Real Penalties (*EQRP*) a new network-constrained trajectory distance that combines warping and node-to-trajectory distances.

The remaining of the paper is structured as follows: to support our motivations, we start by explaining some notions related to network-constrained trajectories. Then we present the related works. After, we define our new distance *EQRP* and describe the methodology adopted for the comparison of some state-of-art network-constrained distances to the newly proposed one. Later, we present the implementation details of our experiments, the results obtained and a discussion of these results before concluding.

II. Definitions

A road network is a set of interconnected road segments that support mobility in a given geographical space. Usually, it is modeled as a graph $G(V, E)$ where V represents the set of nodes which are often road intersections and E corresponds to the set of edges which are road segments linking the nodes.

A trajectory is a time-ordered sequence of geographic positions, usually identified by latitudes and longitudes, which describe a movement. In the context of a network-constrained environment such as an urban road network, the geographic positions can be replaced by the nodes of the graph representing the road network. In the rest of the paper, we consider trajectories as sequences of road network nodes. We formally define a trajectory T as:

$$T = ((v_1, t_1), ..., (v_i, t_i), ..., (v_n, t_n)) \quad (1)$$

Where $v_i \in V$ are nodes and $t_1 < ... < t_i < ... < t_n$ are timestamps associated to nodes. Timestamps can be omitted in various situations.

III. Related work

Although trajectory data analysis has been an active research field for decades, similarity/distance measures specifically designed for network-constrained trajectories emerged only recently. One of the first distance integrating road network structure in its definition was proposed in 2005 by Hwang et al. [1]. It compares trajectories according to a set of Points Of Interest (POIs) which can be road intersections or places. Two trajectories are considered similar if all the POIs lie on both of them otherwise there are not. Thus, this distance measure, which does not respect triangle inequality, can only take two values 0 or 1, and is not able to compare trajectories accurately. Moreover, POIs must be predefined by users who may not have extended knowledge of the road network. Following this first distance, Tiakas et al. [2] presented a distance for network-constrained trajectories, similar to the Euclidean distance for free-space trajectories. This distance corresponds to the average shortest path distances between nodes of same index (i.e nodes that appear at the same position in compared trajectories). It respects triangle inequality, but cannot handle local time shifting or trajectories with different lengths.

To ensure that, no matter the size of trajectories, it is always possible to compute a network distance between them, Evans et al. [3] opted for a node-to-trajectory approach. Here, the distance between two trajectories is equal to the average of minimum distances between each node of one trajectory and all the nodes of the second. In the same vein, [4] defined a spatio-temporal similarity measure by converting the node-to-trajectory distance to a similarity measure using an exponential function. This second measure has been labeled as Spatio-temporal Linear Combine *STLC* in [5]. Notice that both node-to-trajectory measures do not respect triangle inequality.

Another node-to-trajectory distance is the *Hausdorff* distance. It measures the mutual proximity of two sets, by indicating the maximal distance from any node in one set to the other set. [6] and [7] proposed versions of *Hausdorff* distance adapted to network constrained trajectories by replacing Euclidean distance with network distance. The version in [6] respect triangle inequality contrary to the one in [7].

Besides, we also have distances based on the intersection of sets. They are used for applications like carpooling or network flow analysis where similar trajectories are those which share common road segments. Usually, these distances are computed edge-wise to ease the detection of common road segments and to integrate the length of road segments. As examples, we mention *SIM_TRAJ* [8], a similarity that equals to the ratio between the length of two trajectories' common parts and the length of one of them selected as reference, *DSL* [9] a dissimilarity which corresponds to the length of two trajectories uncommon parts over the total length of all road segments forming them and *EBD* [10] equivalent to the length of the longest trajectory minus the length of the compared trajectories' intersection. Among the three, only *EBD* respect triangle inequality.

The previously presented distances are all sequence-

independent, i.e. they don't consider the order of trajectories' nodes, contrary to warping measures. Warping measures for network-constrained trajectories are mainly extensions of measures for free-space trajectories such as: Longest Common Subsequence (*LCSS*) [11], Dynamic Time Warping (*DTW*) [12], Edit Distance on Real sequence (*EDR*) [13] and Edit distance with Real Penalty (*ERP*) [14]. They are sequence-dependent and able to handle local time shifting and trajectories with different lengths. Shang et al. [15] proposed the first warping measure adapted to network-constrained trajectories. Their measure extends *LCSS* which compares two trajectories by determining the size of their longest common subsequence. The points constituting this sub-sequence must not necessarily occupy consecutive positions but must appear in the same order within the two trajectories. The particularity of *LCSS* is to match two points, even if they do not coincide, as long as the distance between them is below a threshold. Later, [16] defined *LORS* (Longest Overlapping Road Segments) which is also a variation of *LCSS* using the sequence of edges. In addition to the trajectory modeling difference, *LORS* matches edges only when they are identical while the former measure depends on a threshold value to loosely match nodes. Then, [17] built *LCRS* (Longest Common Road Segments) on top of *LORS* using Jaccard index. Note that the idea of using the Jaccard index to define trajectory similarity was first presented by Xia et al. in [18]. Network versions of *DTW* [19], *EDR* and *ERP* [20] were also proposed.

These last three distances have the advantage to be complete match warping measures, i.e. all the nodes of compared trajectories are used in the calculation of the distance, contrary to *LCSS* based distances which are partial match distances. A match corresponds to a pairing between two nodes, each belonging to one of the compared trajectories. The distance between these matching nodes is used to compute the global distance between the trajectories. Concerning *DTW*, its particularity is to allow sequence stretching or shrinking in time by replicating previously processed points when there is no match. This ensures that there is always a matching pair for any trajectory point. *EDR* and *ERP* are edit distances that evaluate the cost of edit operations namely, insertion, deletion, and substitution necessary to transform a trajectory into another. Substitution corresponds to a match while insertion and deletion occur when there is no match which is also called a gap. The cost of each edit operation is designated as a subcost.

Despite their qualities, edit distances also bear shortcomings. *EDR*, for example, depends on a matching threshold which is often difficult to fix. It also uses fixed subcosts that are not correlated to the actual distances between nodes. This negatively impacts the ability to accurately assess the distances between trajectories. *ERP*, for its part, is defined according to real distances between nodes but requires the specification of a reference node in the evaluation of the distances between trajectories to handle gaps. The impact of this reference node on *ERP* accuracy in graph space had not been studied. Moreover, the use of real distances between nodes makes *ERP* sensitive to the presence of deformations in the trajectories. Among warping distances, only *ERP* respect triangle inequality.

As a wrap-up, we retain that some network-constrained trajectory distances consider trajectories only as non-ordered sets while others treat them only as sequences although trajectories bear both properties. Moreover, despite their respective qualities, none of these distances concomitantly respect triangle inequality, is parameter-free, and able to manage local time shifting, even though each of these characteristics is important to guarantee a good comparison of trajectories. These are the reasons that led us to define a new measure that meets all these needs.

IV. Edit Distance with Quasi Real Penalties

The measure we propose differs from existing edit distances by how it handles gaps during the matching process. More precisely, when there is a match, the classical network distance between pair of nodes is used. However, if there is a gap, the insertion or deletion cost is fixed at the *Hausdorff* distance between the compared trajectories. Let's recall that *Hausdorff* distance measures the mutual proximity of two sets, by indicating the maximal distance from any node in one set to the other set. Doing so, we replace the fixed cost of *EDR* (always equal to 1) by a semi-fixed cost, i.e. which is fixed only for a pair of trajectories. This also allows us to avoid the use of a reference node contrary to *ERP*. The result is an edit distance that considers trajectories both as sequences and non-ordered sets of nodes (thanks to *Hausdorff* distance). This definition also makes it possible to indirectly set the matching threshold between nodes to the *Hausdorff* distance between trajectories. The new edit distance we introduce is named Edit distance with Quasi Real Penalties (*EQRP*). It is formally defined between two trajectories T_1 and T_2 of respective sizes n and m by the recurrence formula:

$$EQRP(T_1,T_2) = \begin{cases} 0, \ \textit{If } n = 0 \textit{ and } m = 0 \\ \infty, \ \textit{If } n = 0 \textit{ or } m = 0 \\ min \begin{cases} EQRP(Rest(T_1), Rest(T_2)) + d_G(Head(T_1), Head(T_2)) + D \\ EQRP(Rest(T_1), T_2) + Hausdorff(T_1, T_2) + D \\ EQRP(T_1, Rest(T_2)) + Hausdorff(T_1, T_2) + D \end{cases} & \textit{Otherwise} \end{cases} \tag{2}$$

$Head$ designates the first node of the considered sequence, $Rest$ refers to the whole sequence except the first node and d_G is the network distance between two nodes, and D is the diameter of the network. The *Hausdorff* distance formula is :

$$Hausdorff(T_1,T_2) = max \begin{cases} max_{v_i \in T_1} min_{v_j \in T_2} d_G(v_i, v_j) \\ max_{v_j \in T_2} min_{v_i \in T_1} d_G(v_i, v_j) \end{cases} \tag{3}$$

Adding the diameter to the different subcosts ensures that *EQRP* respects the triangle inequality: for any triplet of distinct trajectories R, S, and T we have

$$EQRP(R,S) \leq EQRP(R,T) + EQRP(T,S) \tag{4}$$

Indeed, for an edit distance to respect this inequality, it is sufficient that the distance between elements on which it is based also respects it even if there is a gap [21]. When there is

no gap, we use the graph distance (the shortest path between nodes) which respects the triangle inequality. Conversely, if there is a gap, the inequality remains respected because the *Hausdorff* distance is necessarily smaller than or equal to D.

As other warping distances, ***EQRP*** can be calculated using a dynamic programming algorithm to solve its recurrence formula. This method of calculation has a complexity of $\mathcal{O}(nm(|V|+|E|))$ with n and m, the respective sizes of the compared trajectories and $(|V|+|E|)$ the cost of computing the shortest path between two nodes. This complexity includes the cost of pre-calculating the *Hausdorff* distance whose cost is also $\mathcal{O}(nm(|V|+|E|))$.

V. Comparison to other distances

After having formally defined our new distance, it is important to ensure its relevance by comparing it to existing network-constrained distances. However, comparing distances is not trivial because the values returned by different distances for the same pair of trajectories are not always consistent as each distance has its own semantics. Therefore, it is not appropriate to rely directly on the values and we need to proceed through indirect approaches. In this section, we describe two alternative approaches: resistance to transformations and clustering ability.

A. Comparison by transformations

The method of comparison by transformations was initially proposed by Han et al in their review [5]. It aims to compare distances between free-space trajectories by evaluating their ability to resist different transformations: addition or deletion of sampling points; trajectory resampling; time dilation and compression; spatial expansion and compression; noise addition. Here, we do not explicitly take into account the temporal data of the trajectories, because most of the compared distances do not integrate them, so the transformations relating exclusively to the time domain are discarded (time dilation and compression, trajectory resampling). Additionally, the trajectories are modeled as sequences of nodes, it is subsequently not possible to add sample points without modifying the road network. The constraints of the road network also make it very difficult to expand or compress the trajectories without changing their shape. Due to these limitations, we retain only two of the transformations initially proposed and adapt them to network-constrained trajectories. We renamed the deletion of sample points as nodes deletion and we decline the noise addition in two versions: loops and detours addition.

1) Nodes deletion: Before any node deletion, we identify first the pivot nodes which are nodes where the trajectory noticeably changes shape. This is done using the Ramer-Douglas-Peuchker (RDP) algorithm [22] whose principle is to replace a polyline (a broken line) made up of several points by a direct line, i.e. a shortcut, between the endpoints. This modification is only applied if the distance between the direct line and the furthest point of the polyline is less than a predefined threshold. To adapt this idea to network-constrained trajectories, we add the GPS coordinates as attributes to the nodes. Once pivot nodes are identified, we randomly delete some nodes among those remaining. The number of nodes removed depends on a ratio r whose values are between 0 and 1. For a trajectory T, of size n, with k pivot nodes, the number of nodes to remove is $NB_{pts} = \min\{n * r, n - k\}$. The min ensures that only the non-pivot nodes can be deleted.

2) Loops and detour addition: In free-space, noise addition to a trajectory results in adding Gaussian noise to points' coordinates, while for network-constrained trajectories, it corresponds to the addition of deviations in the route of the trajectories. The reason is that the space of the road network is discrete, which prevents the addition of points other than those present in the set of nodes forming the network. Two types of deviations can be generated: a loop or a detour. A loop is a circuit in the road network, a sequence of nodes starting and ending with the same node. A detour is an alternative route between two distinct nodes of the same trajectory. Usually, these two disturbances occur during the map-matching process when the raw trajectories contain noise or due to an error in the matching process between points and road segments. The objective here is to try to reproduce them artificially.

To add a loop, we randomly choose a node u in the trajectory and a node v outside the trajectory but within a predefined radius around u. We calculate the shortest paths from u to v and vice-versa. The resulting path, which forms a loop, is then inserted in the trajectory: once arrived at u, we go through the loop before continuing the initial trajectory. The number of loops to add to a trajectory is also managed by a ratio between 0 and 1.

The procedure to add a detour is similar. First, we choose two distinct nodes u and v of the same trajectory separated by a given number of nodes p (for projection). Then, we select a random node w outside the trajectory but accessible from u within a predefined radius. Then we determine the shortest path going from u to v via w and we replace the sub-sequence of the original trajectory between u and v by the one passing through w. Note that the detour is not necessarily longer than the path it replaces in the original trajectory, because the sub-trajectory from u to v had no reason to be the shortest path between these two nodes. The detour is therefore just an alternative path passing through the vertex w.

3) Evaluation of transformations impact: To quantify the robustness of distances against transformations, we rely on the method described in [5]. First, we randomly select a reference trajectory from the original dataset. Second, we extract the ordered list of trajectories closest to the reference trajectory, using the distance we want to evaluate, sorted by increasing value. This list is the *original_list*. Third, we apply a given transformation to all trajectories in the original dataset apart from the reference trajectory, to obtain a derived dataset. We then order the trajectories of the derived dataset by increasing distance to the reference (and untouched) trajectory to obtain a *transformed_list*. Fourth, we evaluate the similarity between the *original_list* and the *transformed_list*. This is repeated for each transformation. The intuition behind this approach is that a distance that resists well to transformations will tend to

similarly order the closest trajectories to a reference trajectory, both in the original and in derived datasets. The similarity between the ordered lists is computed using Spearman's rank correlation coefficient [23] that measures the correlation between two rankings X and Y of the same size as:

$$r_s = \frac{cov(X,Y)}{\sigma_X.\sigma_Y} \tag{5}$$

With cov the covariance and σ the standard deviation. Spearman's measure takes values from -1 to 1. A value close to 1 means that the rankings are similar which also indicates a good robustness of the distances against transformations. A value close to 0 indicates an absence of link between the two rankings and therefore a low, or non-existent, resistance of distances against transformations. A value close to -1 indicates a decreasing trend: close trajectories in the original data are distant in the derived data and vice-versa.

Comparing distances through correlations can be a source of bias in the case of measures such as *EDR* and *LCSS* because of the way they are calculated. Indeed, these measures consider any pair of trajectories having no matching pair of nodes as being at a maximum distance from each other, i.e. at distance 1 when distances are normalized. This poses a problem of resolution in the trajectories comparison and biases the nearest neighbor ordering. More specifically, we note that from a certain index, the order of the nearest neighbors does not vary anymore since all the trajectories are found at distance 1 from the reference trajectory regardless of the transformations. We then obtain high correlation values that do not reflect the ability of the distance to manage a transformation but rather its inability to do so. We could possibly choose to exclude the trajectories at a maximum distance but this problem also arises in a more general way when distances can only take a limited set of values, small compared to the number of trajectories, and that many trajectories are therefore at the same distance from the reference trajectory. To overcome this bias, we break the equality of the trajectories located at the same distance from the reference trajectory by a random choice. For a better understanding, consider six trajectories $T2$; $T3$; $T4$; $T5$; $T6$; $T7$ whose respective distances to the reference trajectory $T1$ are: 0.3; 1; 0.5; 1; 1; 0.2. The natural ordering of the nearest neighbors of $T1$ would be $T7$; $T2$; $T4$; $T3$; $T5$; $T6$ if we used the order of the indices in case of equality. If the distance used has a limited resolution as explained above, the order of the last three trajectories $T3$; $T5$; $T6$ has very little chance to change because of transformations. We break this constant ordering by randomly ranking $T3$; $T5$; $T6$ each time they are all at the maximum distance from the reference trajectory. In an extreme case, if all the trajectories are at distance 1 from the reference trajectory in the original dataset and in a derived dataset, this solution will lead to a total lack of correlation between the two rankings. For the sake of impartiality, we apply this random ranking technique to all distances.

Comparison of distances according to their robustness to transformations does not assess their resolution, i.e. their ability to accurately account for the relative differences between trajectories. In order to better assess this resolution, we propose to also use a comparison by clustering.

B. Comparison by Clustering

A way to compare network-constrained trajectory distances is to measure how well they can cluster trajectories. The underlying idea is that if we have groups of similar trajectories (visually close for example) then the more discriminating a distance is, the better it should distribute the trajectories in their original clusters. To compare distances by clustering, we select a sample of trajectories that we divided into distinct groups. Each group consists of trajectories that are geographically close to each other and the clusters do not overlap. Trajectories are clustered using Hierarchical Cluster Analysis (*HCA*) algorithm. The comparison of the clustering results is done using the Adjusted Rand Index (ARI) measure [24] and the Silhouette coefficient [25].

ARI estimates the quality of clustering by comparing the result obtained with the ground truth while ignoring permutations. Indeed, it may happen that the trajectories are perfectly distributed in clusters but the order of these clusters differs from that provided by the ground truth. Such a result must be counted as correct, hence the need to ignore permutations between clusters. The other specificity of ARI measure is that random clustering returns a value close to 0 regardless of the dataset size and the number of clusters. The formula of the ARI measure is:

$$ARI = \frac{RI - E[RI]}{max(RI) - E[RI]} \tag{6}$$

where RI is the Rand Index measure from which ARI is derived, $RI = (a+b)/\binom{n}{2}$; a is the number of pairs of elements that belong to the same cluster in the ground truth and in the result of the clustering; b is the number of pairs of elements that belong to the same cluster in the ground truth and to different clusters in the clustering result; $E[RI]$ is the expectation of the Rand Index and is obtained by calculating this measurement on random distributions of elements in clusters.

The Silhouette coefficient measures the similarity of an object with its own group compared to other groups. It does not require a ground truth and is defined as:

$$S = \frac{b-a}{max(a,b)} \tag{7}$$

With a the average distance between an element and all the other members of its cluster and b the average distance between an element and all the elements of the closest cluster. The Silhouette coefficient for all the elements of a dataset corresponds to the average of the Silhouette coefficients of the different elements. The values of the ARI measure and the Silhouette coefficient range from -1 (worst case) to 1 (best case). Negative values usually indicate the misassignment of elements to clusters. Values close to 0 indicate clusters overlapping for the Silhouette coefficient and random assignments for the ARI measure. The advantage of the Silhouette

coefficient compared to the ARI measure is that it does not require ground truth.

VI. Experimentation

Experiments were performed on a server running UBUNTU 16.04. Geographic data are processed with Quantum GIS 3.4 Madeira (*QGIS 3.4*).

A. Road network and trajectories

The road network is extracted from the map of the administrative region of Porto (Portugal) downloaded on OpenStreetMap[1] and using the open-source tool Osm2Po[2]. The directed graph obtained is strongly connected with 350189 nodes representing intersections or road ends and 811583 edges.

Trajectories were extracted from Porto trajectory dataset [26] which contain 1710670 raw trajectories corresponding to 442 taxis' journeys in the city of Porto and its surroundings over a whole year, from 01/07/2013 to 30/06/2014. Each trip is described by an identifier, a set of GPS coordinates measured every 15 seconds, and various additional information about the taxi, the origin of the trip request, etc. The trajectories selected have been map-matched using barefoot[3], an open-source library written in Java, and implementing the Hidden Markov Model (HMM). Each trajectory is then transformed into a sequence of road network nodes.

B. Distances

We compare *EQRP* to network versions of *STLC*, *Hausdorff* distance, *DTW*, *LCSS*, *EDR* and *ERP* in terms of clustering performances and transformations handling. Table I presents the formulas of the competing distances network versions which all have a complexity of $\mathcal{O}(nm(|V|+|E|))$. Sizes of trajectories T_1 and T_2 are respectively n and m. $Head$ designates the first node of the considered sequence, $Rest$ refers to the whole sequence except the first node, v_g is the reference node, and d_G is the network distance between two nodes. To implement the network version of *ERP*, we had to define a reference node for the case when there is a gap. Among diverse possibilities, we choose the reference node based on the criterion of eccentricity. Recall that the eccentricity of a vertex is the maximum distance between this vertex and all the other vertices of the graph. According to this choice, two variants of the graph version of *ERP* were implemented. In the first variant (ERP_{G1}), the reference node is one of those having the minimum eccentricity, and in the second (ERP_{G2}) it is a node of maximum eccentricity which is considered. A node of minimum eccentricity naturally constitutes a reference in a graph since it belongs to its center. Likewise, a maximum eccentricity node can act as a reference node because it is the most eccentric of all. Concerning *LCSS* and *EDR*, the value of the threshold ϵ is set at 0 because the imprecision associated with the geographic coordinates had been removed by the map-matching operation. All the distances are implemented in python. We also use python implementation of hierarchical clustering provided by *SciPy*[4] library.

C. Transformations parameters

We choose to delete 20% of vertices in the nodes deletion transformation. Concerning the loops addition transformation, we select 5% of nodes in each trajectory and set the neighborhood radius to 500 meters. For the detours adding transformation, we set the projection parameter to 5 (nodes) in order to simulate medium detours. The neighborhood radius parameter remains fixed at 500 meters. These parameters make it possible to limit the number of generated loops or detours on the trajectories in order to avoid strong distortions. This choice of neighborhood radius aims to produce loops of intermediate size (neither too small nor too large) and also ensure the presence of neighboring nodes in the graph representing the road network. Note that we conducted various experiments with different parameters and the results obtained follow the same tendency as those corresponding to the parameters mentioned above.

D. Clustering

The ground truth for the clustering consists of a sample of 100 trajectories split into five non-overlapping clusters of 20 trajectories each. Figure 2 illustrates the disjoint zones as well as the trajectories that are extracted there. We compute the pairwise distance matrices for each distance and apply *Hierarchical Cluster Analysis*. We use the ward linkage measure for *HCA* to minimize the variance of clusters being merged.

Fig. 2: Distinct clusters with their trajectories.

VII. Results

A. Impact of nodes deletion

Figure 3 presents the results for nodes deletion. Among the distances compared, *Hausdorff* appears to be the most robust, with correlations regularly between 80% and more than 90%. *STLC* is the second distance that best handles the effects of nodes deletion transformation with correlation

[1]https://www.openstreetmap.org
[2]https://osm2po.de
[3]https://github.com/bmwcarit/barefoot
[4]https://docs.scipy.org

TABLE I: Competing distances formulas

Distances	**Formulas**
$STLC(T_1, T_2)$	$= \frac{\sum_{v_i \in T_1} e^{-d_G(v_i, T_2)}}{n} + \frac{\sum_{v_j \in T_2} e^{-d_G(v_j, T_1)}}{m}$ $d_G(v_i, T_2) = \min_{v_j \in T_2} d_G(v_i, v_j)$
$DTW(T_1, T_2)$	$= \begin{cases} 0, \textit{ If n = 0 and m = 0} \\ \infty, \textit{ If n = 0 or m = 0} \\ d_G(Head(T_1), Head(T_2)) + min \begin{cases} DTW(T_1, Rest(T_2)) \\ DTW(Rest(T_1), T_2) \\ DTW(Rest(T_1), Rest(T_2)) \end{cases} & \textit{Otherwise} \end{cases}$
$LCSS(T_1, T_2)$	$= \begin{cases} 0, \textit{ If n = 0 or m = 0} \\ 1 + LCSS(Rest(T_1), Rest(T_2)), \textit{ if } d_G(Head(T_1), Head(T_2)) \leq \epsilon, \\ max \begin{cases} LCSS(Rest(T_1), T_2) \\ LCSS(T_1, Rest(T_2)) \end{cases} \textit{Otherwise} \end{cases}$
$EDR(T_1, T_2)$	$= \begin{cases} n, \textit{ If m = 0} \\ m, \textit{ If n = 0} \\ min \begin{cases} EDR(Rest(T_1), Rest(T_2)) + subcost \\ EDR(Rest(T_1), T_2) + 1 \\ EDR(T_1, Rest(T_2)) + 1 \end{cases} & \textit{Otherwise} \end{cases}$ $subcost = \begin{cases} 0, \textit{ If } d_G(Head(T_1), Head(T_2)) \leq \epsilon \\ 1, \textit{ Otherwise} \end{cases}$
$ERP(T_1, T_2)$	$= \begin{cases} \sum_l^n d_G(v_i, v_g), \textit{ If m = 0} \\ \sum_l^m d_G(v_j, v_g), \textit{ If n = 0} \\ min \begin{cases} ERP(Rest(T_1), Rest(T_2)) + d_G(Head(T_1), Head(T_2)) \\ ERP(Rest(T_1), T_2) + d_G(Head(T_1), v_g) \\ ERP(T_1, Rest(T_2)) + d_G(Head(T_2), v_g) \end{cases} & \textit{Otherwise} \end{cases}$

values regularly above 80%. *EQRP* proves its robustness by generating correlations around 70 % in the best case. Moreover, it is much more efficient than the graph versions of the other warping distances as *ERP*, *EDR*, *DTW* and *LCSS*. Only *DTW* produce average correlation values while the other three have correlations close to 0. The performance of *ERP*, regardless of the variant considered, stands out as the worst.

B. Impact of adding loops

The curves presented in Figure 4 show that in the case of loop addition, the three best distances are in order *Hausdorff*, *STLC* and *EQRP*. The *Hausdorff* distance obtains the best performance, with correlations reaching 80% while *STLC* correlation peak is between 70% and 80%. The curves describing the evolution of the *EQRP* correlations indicate a good capability to resist the addition of loops with correlation values that reach 70%. The graph versions of *ERP*, *EDR*, *LCSS* and *DTW* remain less efficient than the leading trio. Especially *ERP* correlations are close to 0 as for the nodes deletion transformation. However *DTW* stands out with better correlation values that reach almost 50%. This gives it a medium resistance to transformation by adding loops.

C. Impact of adding detours

From Figure 5 we note that adding detours has quite a similar impact on distances as loops addition. That could be expected as loops are kind of detours. *ERP* variants are again the least resilient to this transformation with correlations close to 0. *EDR* and *LCSS* follow respectively with correlation also close to 0, which also show a weak capacity to manage this transformation. *DTW*, presents an average resistance with correlations reaching at best 50%. The three best distances remain in order: *Hausdorff*, *STLC* and *EQRP*. *Hausdorff* distance and *STLC* present performances that reach 80%, while *EQRP* reach 70%. This shows their good ability to manage the addition of detours.

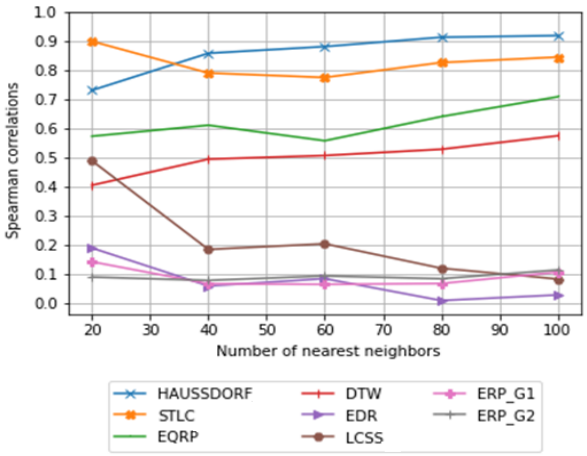

Fig. 3: Spearman's correlations by distance as a function of the number of nearest neighbors for the nodes deletion transformation (Ratio = 20%).

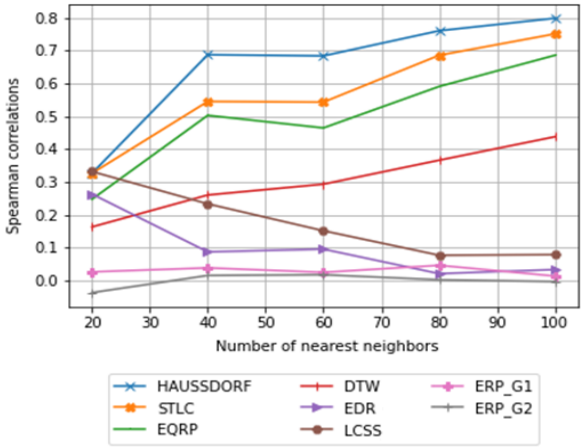

Fig. 4: Spearman's correlations by distance as a function of the number of nearest neighbors for the adding loop transformation (Ratio = 5% & Radius = 500 meters).

D. Clustering performances

The results from the clustering comparison, presented in table II, corroborate the conclusions resulting from experiences related to trajectory transformations. *Hausdorff* distance, *STLC* and *EQRP* manage to distribute perfectly (or almost perfectly) trajectories into distinct clusters. The same goes for *DTW* which stands out here, with substantially equal performance unlike what was observed for the comparison by transformations. *LCSS*, *EDR* and *ERP* variants produce low quality partitioning. *ERP* remains the worst-performing distance.

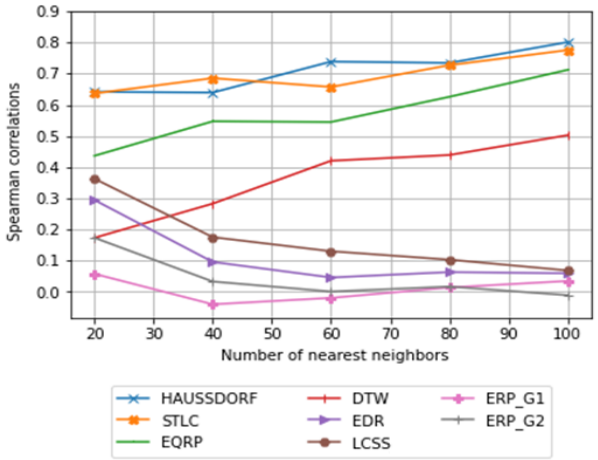

Fig. 5: Spearman's correlations by distance as a function of the number of nearest neighbors for the adding detours transformation (Projection = 5 & Radius = 500 meters).

TABLE II: Clustering performances by distance

Distances	ARI	Silhouette
$STLC$	1.0	0.887
DTW	1.0	0.557
$EQRP$	1.0	0.538
$Hausdorff$	0.975	0.488
$LCSS$	0.283	0.147
EDR	0.173	0.106
ERP_{G1}	0.022	0.367
ERP_{G2}	0.006	0.434

VIII. Discussion

The *Hausdorff* distance resists well to various transformations because of its formulation based on the evaluation of the minimum distances between nodes and trajectories. Actually, these distances tend to remain stable even when the trajectories are deformed. Indeed, the minimum distance between a node u and a trajectory varies only if the trajectory noticeably changes its shape so that it modifies its node closest to u. *STLC* good abilities can also be explained by the node-to-trajectory distance computation scheme. The case of *DTW* is particular in the sense that it is a relatively sensitive distance against transformations but it has a good discrimination capability. This is explained by the fact that *DTW* computes the real distances between the nodes of the compared trajectories. It subsequently returns a relatively precise distance between trajectories but any modification in the position of the nodes is automatically reflected in the distance calculated. Concerning warping distances *LCSS* and *EDR*, their weak performance is due to their calculation method that does not include the actual distance between nodes and uses matching thresholds. In the case of *ERP* we suspect a negative impact of the reference nodes even if it remains to be proved. Our choice of a double implementation of *ERP* network version, aimed to highlight the impact of the reference node choice on its

robustness and accuracy but the results do not lead to a clear conclusion. Although the results seem to show a slight predominance of *Hausdorff* distance and *STLC* over *EQRP* regarding transformations, it should be remembered that these two distances do not take into account the order of the nodes and are therefore limited in their evaluation of the distances between trajectories.

This limitation effect is mainly perceptible during the comparison to similar trajectories. Take the example of three car trajectories T_1, T_2 and T_3 such that the first two are disjoint and T_3 is identical to T_2 in terms of nodes but their sequences are reversed, i.e. T_2 is from node A to node B while T_3 is in the opposite direction. When comparing T_1 to T_2 and T_3 with *Hausdorff* distance and *STLC*, it is not possible to distinguish T_2 from T_3 which will appear perfectly identical because they are at the same distance of T_1. However *EQRP* will highlight the difference due to the direction of each trajectory because it aligns trajectories according to their nodes' sequences.

IX. Conclusion

We propose a new distance for network-constrained trajectories combining the qualities of node-to-trajectory distances with those of the edit distances. This distance also considers trajectories both as non-ordered sets and as sequences. We compare our distance to some of the most popular distances for network-constrained trajectories by assessing their resistance to transformations and their clustering performances. The result shows that our distance outperforms all the existing edit distances and is in some cases more effective than node-to-trajectory distances.

As future work, we are interested in the use of *EQRP* to query large network-constrained trajectories databases. We also project to create a network-constrained trajectory analysis platform integrating a search engine. This is a perspective that would lead to the development of parallel or distributed architectures for trajectories storage and distances computation and will include indexes specifically defined to the *EQRP* distance, based for example on the triangle inequality.

References

[1] J.-R. Hwang, H.-Y. Kang, and K.-J. Li, "Spatio-temporal similarity analysis between trajectories on road networks," in *International Conference on Conceptual Modeling*. Springer, 2005, pp. 280–289.

[2] E. Tiakas, A. Papadopoulos, A. Nanopoulos, Y. Manolopoulos, D. Stojanovic, and S. Djordjevic-Kajan, "Searching for similar trajectories in spatial networks," *Journal of Systems and Software*, vol. 82, no. 5, pp. 772–788, 2009.

[3] M. R. Evans, D. Oliver, S. Shekhar, and F. Harvey, "Summarizing trajectories into k-primary corridors: a summary of results," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, 2012, pp. 454–457.

[4] S. Shang, L. Chen, Z. Wei, C. S. Jensen, K. Zheng, and P. Kalnis, "Trajectory similarity join in spatial networks," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, 2017.

[6] G.-P. Roh and S.-w. Hwang, "Nncluster: An efficient clustering algorithm for road network trajectories," in *International Conference on Database Systems for Advanced Applications*. Springer, 2010, pp. 47–61.

[5] H. Su, S. Liu, B. Zheng, X. Zhou, and K. Zheng, "A survey of trajectory distance measures and performance evaluation," *The VLDB Journal*, vol. 29, no. 1, pp. 3–32, 2020.

[7] M. R. Evans, D. Oliver, S. Shekhar, and F. Harvey, "Fast and exact network trajectory similarity computation: a case-study on bicycle corridor planning," in *Proceedings of the 2nd ACM SIGKDD international workshop on urban computing*, 2013, pp. 1–8.

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

[9] J.-I. Won, S.-W. Kim, J.-H. Baek, and J. Lee, "Trajectory clustering in road network environment," in *2009 IEEE Symposium on Computational Intelligence and Data Mining*. IEEE, 2009, pp. 299–305.

[10] S. Wang, Z. Bao, J. S. Culpepper, T. Sellis, and X. Qin, "Fast large-scale trajectory clustering," *Proceedings of the VLDB Endowment*, vol. 13, no. 1, pp. 29–42, 2019.

[11] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proceedings 18th international conference on data engineering*. IEEE, 2002, pp. 673–684.

[12] D. J. Berndt and J. Clifford, "Finding patterns in time series: a dynamic programming approach," in *Advances in knowledge discovery and data mining*, 1996, pp. 229–248.

[13] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 2005, pp. 491–502.

[14] L. Chen and R. Ng, "On the marriage of lp-norms and edit distance," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, 2004, pp. 792–803.

[15] S. Shang, R. Ding, K. Zheng, C. S. Jensen, P. Kalnis, and X. Zhou, "Personalized trajectory matching in spatial networks," *The VLDB Journal*, vol. 23, no. 3, pp. 449–468, 2014.

[16] S. Wang, Z. Bao, J. S. Culpepper, Z. Xie, Q. Liu, and X. Qin, "Torch: A search engine for trajectory data," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 535–544.

[17] H. Yuan and G. Li, "Distributed in-memory trajectory similarity search and join on road network," in *2019 IEEE 35th international conference on data engineering (ICDE)*. IEEE, 2019, pp. 1262–1273.

[18] Y. Xia, G.-Y. Wang, X. Zhang, G.-B. Kim, and H.-Y. Bae, "Spatio-temporal similarity measure for network constrained trajectory data," *International Journal of Computational Intelligence Systems*, vol. 4, no. 5, pp. 1070–1079, 2011.

[19] Z. Shang, G. Li, and Z. Bao, "Dita: Distributed in-memory trajectory analytics," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 725–740.

[20] S. Koide, C. Xiao, and Y. Ishikawa, "Fast subtrajectory similarity search in road networks under weighted edit distance constraints," *arXiv preprint arXiv:2006.05564*, 2020.

[21] M. S. Waterman, T. F. Smith, and W. A. Beyer, "Some biological sequence metrics," *Advances in Mathematics*, vol. 20, no. 3, pp. 367–387, 1976.

[22] P. S. Heckbert and M. Garland, "Survey of polygonal surface simplification algorithms," Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, Tech. Rep., 1997.

[23] T. D. Gauthier, "Detecting trends using spearman's rank correlation coefficient," *Environmental forensics*, vol. 2, no. 4, pp. 359–362, 2001.

[24] L. Hubert and P. Arabie, "Comparing clusterings," *Journal of Classification*, vol. 2, pp. 193–218, 1985.

[25] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[26] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas, "Predicting taxi-passenger demand using streaming data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1393–1402, Sept 2013.