

Distances in MOPSI

Hatim Oulad Arifi

Master's Thesis



ITÄ-SUOMEN YLIOPISTO

School of Computing

Computer Science

May 2020

UNIVERSITY OF EASTERN FINLAND, Faculty of Science and Forestry, Joensuu School of Computing, Computer Science

Hatim Oulad Arifi: Distance in MOPSI

Master's Thesis, 43 p.,

Supervisors of the master's Thesis: Professor Mr. Pasi Fränti and Dr. Radu Mariescu-Istodor.
April 2020

Abstract: Nowadays, our smartphones can do many useful things for facilitating our life as it proved in the several applications of distances that exist in web stores [23].

In this study, we will try to find an optimal solution for travelling salesman problem that asks to give a list of cities and the distances between each pair of cities, and finding the shortest possible route that visits each city and returns to the origin city using a different classification of distances.

A minimum spanning tree (MST) or minimum weight spanning tree may be a subset of the perimeters of a connected, edge-weighted undirected graph that connects all the vertices together, with none cycles and with the minimum possible total edge weight. It is a spanning tree whose sum of edge weights is as small as possible.

More generally, an edge-weighted undirected graph (not necessarily connected) contains a minimum spanning forest, which can be a union of the minimum spanning trees for its connected components.

The solution for MST contains plenty of information useful for solving TSP still. By modifying the Kruskal algorithm to seek out a heuristic solution for TSP using the geographical distances [24].

Another, solution based on Evolutionary programming can be used for finding the solution to TSP by evolving the numerical value of tours and computing the similarity between them using the concept of distance sequences.

Keywords: Applications of distances travelling salesman problem TSP, minimum spanning tree, Kruskal algorithm, Evolutionary programming

CR Categories (ACM Computing Classification System, 1998 version):

Foreword

This thesis was done at the School of Computing, University of Eastern Finland during the period of my study in Master of Science and computer science.

I would like to give my sincere thanks to University of Eastern Finland for accepting me to the program.

I would also like to thank all the professors who helped my gain knowledge in different areas of computer science.

I would also like to give my deepest gratitude my supervisor, Mr. Pasi Fränti for his guidance during my study.

I would also like to express my deepest gratitude to Dr. Radu Mariescu-Istodor who helps me to make a table of contents and for presenting several technical solutions.

Also, I want to extend my gratitude to the colleagues Himat shah and Lahari Sengupta for answering my questions.

As well, I want to thank my parents and my brother and for the rest of my family for moral support. And Mrs. Olli Kohonen the coordinator of my study for her help and advices.

List of abbreviations

UEF	University of Eastern Finland
MOPSI	Mobile location-based platform for collecting photos and routes
O-MOPSI	Orienteering Mopsi
C-SIM	Cell similarity
RSP	Route Similarity Ranking
OSM	Open street Map
OSRM	Open Source Routing Machine
API	Application programming interface
LCS	Longest common subsequence
TSP	Travelling salesman problem
MST	Minimum spanning tree
MTA	Multiple Tours Alignments
BLAST	Basic local alignment search tool

Table of Contents

1. Introduction	1
2. Geographical distances.....	4
2.1.1 L_p Minkowski distance measures.....	4
2.1.2 Bhattacharyya distance	6
2.1.3 Haversine distance calculation.....	7
2.2 Travel distances.....	10
2.2.1 Road network (OSM / OSRM).....	10
2.2.2 Similarity of routes.....	11
2.2.2.1 C-SIM algorithm	11
2.2.2.2 Route Similarity Ranking (RSR) algorithm.....	12
2.2.2.3 Similarity and inclusion	13
2.2.2.4 Examples	14
2.2.3 Tour length in Mopsi.....	16
3. String distances	19
3.1 Distance between two strings.....	19
3.1.1 Strings of the same length.....	19
3.1.2 Strings of different lengths.....	20
3.1.2.1 Levenshtein distance	20
3.1.2.2 Damerau–Levenshtein distance.....	21
3.2. Distances between sets of strings	22
3.2.1 Longest common subsequence problem	22
3.3.2 String segmentation.....	22
3.3.3 Bag-of-tokens	23
4. Other distances	25
4.1. Multidimensional distances.....	25
4.2. Pairwise distance (Alignment)	26
4.2.1 Dynamic time wrapping.....	26
4.2.2 Matrix distance and construction of the wrapping path	28
4.3 Multiple Tours Alignment and Tours similarity	29
5. Conclusion.....	40
References	41

1 Introduction

Distance is a numerical measurement of how far apart objects or points are. For computing the geographical distance, we need to know the start location and the end location. The location is used to identify a point or an area somewhere. There are two types of location. Relative location describes a displacement of the point from another site. It is called *distance*. An example is "3 kilometers northeast from Joensuu", see Figure 1. Absolute geo-location is defined by exact latitude and longitude ,see Figure 2.



Figure 1. Relative location 3 kilometers northeast from Joensuu [27]

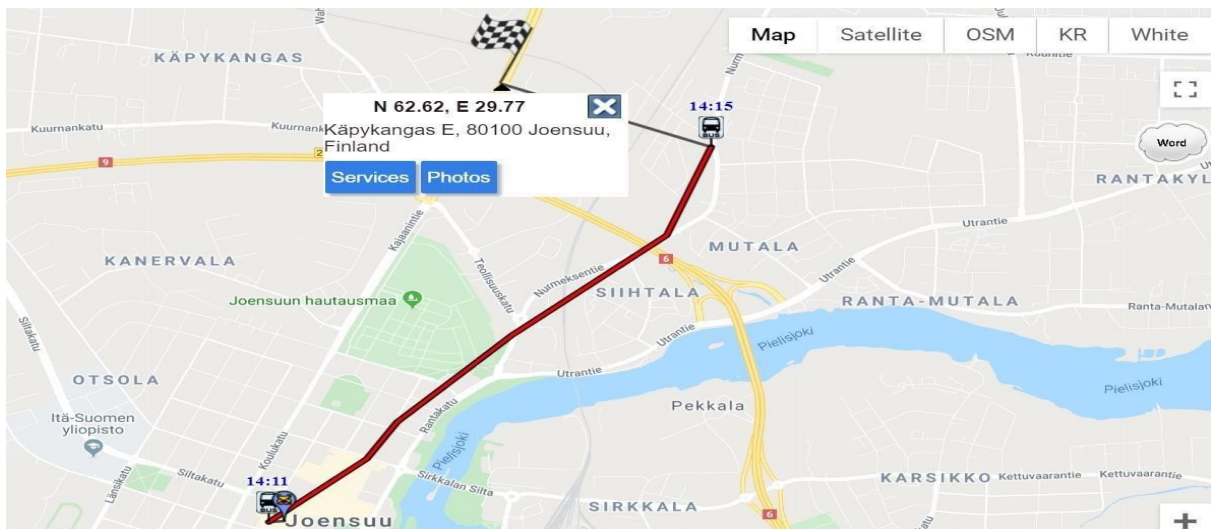


Figure 2. Absolute distance 3 kilometres northwest from Joensuu [27]

Distance can also be given via a sequence of intermediate points. In Figure2, is an example bus route where bus stops define the intermediate points. The concept of distance can be generalized to calculate how much two trajectories differ from each other [1]. However, instead of distance, the concept of similarity is more often used. The shorter the distance between the two objects, the more similar they are.

The similarity measure between two roads is high when the distance is small between the trajectories and the lowest similarity where the distance large. Also, we can consider that two routes are similar if the overlapping happens between them, see Figure 3.

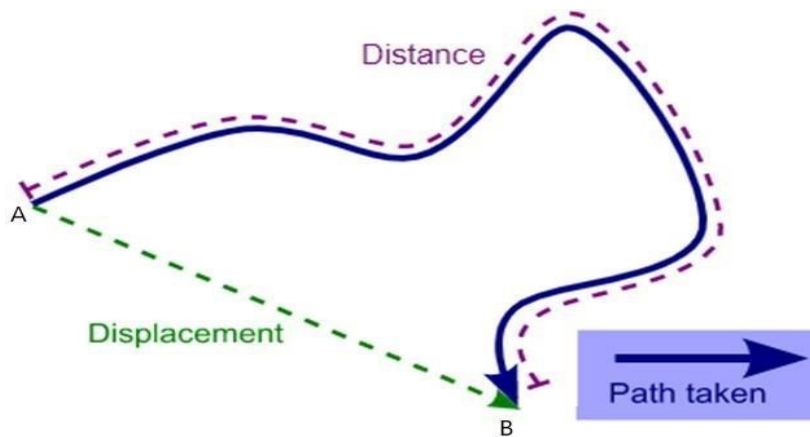


Figure 3. Displacement and distance between locations A and B

Figure 4 shows Mopsi users Pasi, Radu, and Andrei having many similar trajectories in Joensuu and Liperi following a cycling route with different speeds [1], see Figure 4. In Mopsi, trajectory similarity is used to find trajectories that can be compared. Another application of trajectory similarity is to optimize the traffic using a distance function in density-based clustering of segments in order to identify the congestion areas [1]. As well, the applications of the shortest path or the nearest distance and extension to traveling salesman problem and finding the shortest tour.

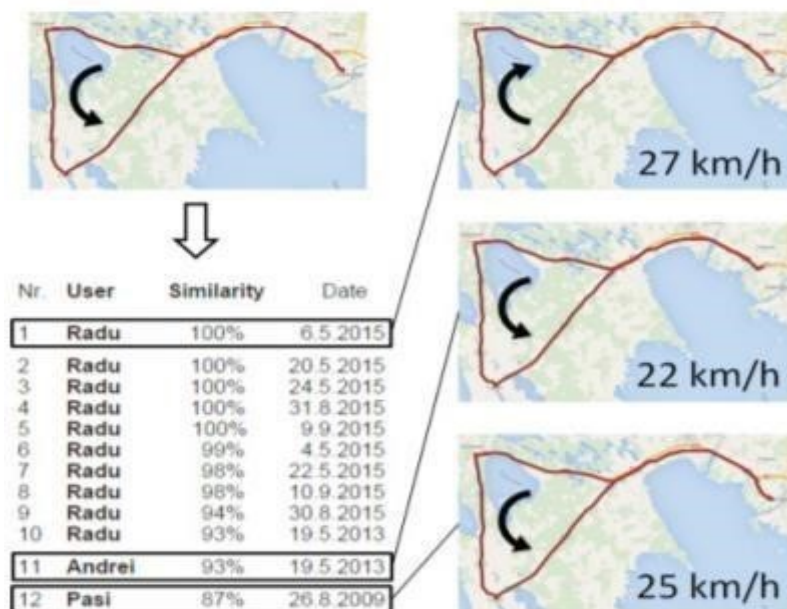


Figure 4. Similarity between two routes [1]

We use the term distance also to measure non-physical entities. For instance, the theoretical distance between two strings where the words may vary by only one letter. For example, "dog" and "dot", differ by only one letter, are closer than "dog" and "cat", see Figure 5. So, if you give the value of

1 for any matching alphabet between the two Sequence a and Sequence b. The string distance between "dog" and "dot" is $d(S_a, S_b) = 2$ and between "dot" and "cat" $d(S_a, S_b) = 1$.

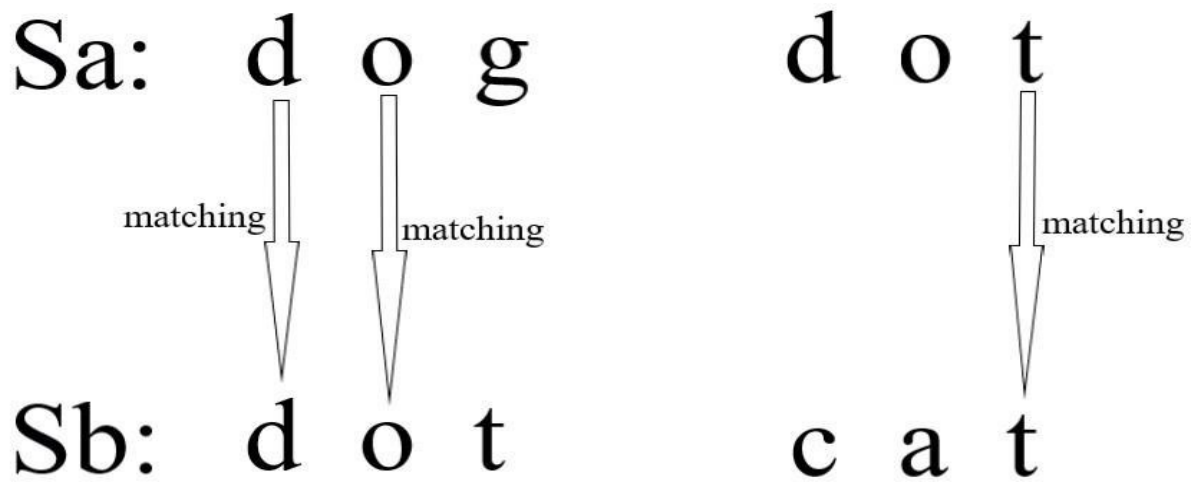


Figure 5. String distance

2. Geographical distances

A physical distance means several different things such as distance travelled on a specific path travelled between two points, or the distance walked while navigating a maze.

2.1 Point distances

There are many methods to calculate the distance between points. The distance between points can be interpreted as the extent of spacing between them.

2.1.1 L_p Minkowski distance measures

In general, the formula of Minkowski distance [8] has a square root that may have nice properties in the distance function.

$$D(p_i, q_i) = \sqrt[p]{\sum |p_i - q_i|^p}$$

We consider the distance between two points A and B where $A=(X_A, Y_A)$, $B=(X_B, Y_B)$ and $C=(X_C, Y_C)$ is their crossing point along the green longitude line. A and C are locations belonging to the red North-south line in Figure 6, it can be called longitude line.

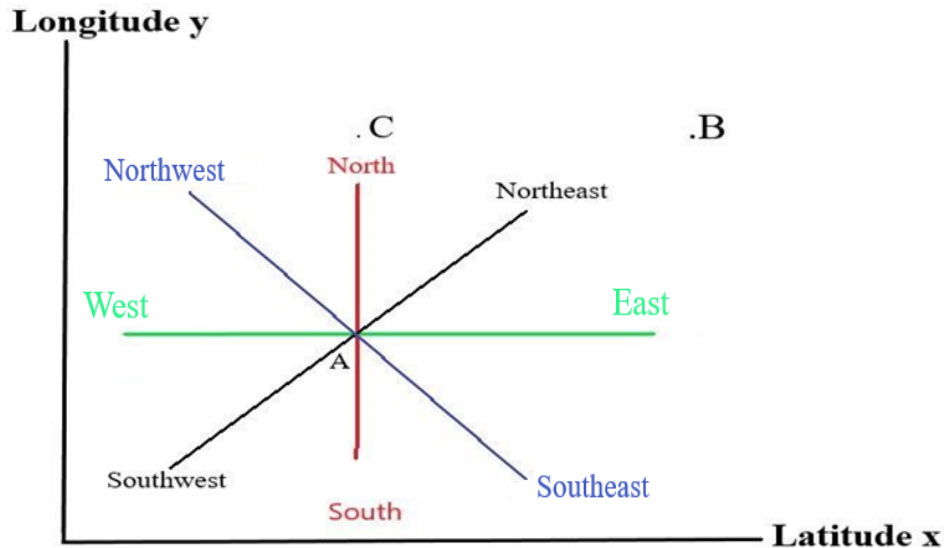


Figure 6. Example of distances from A to B, and A to C.

If $p=2$, we will get the Euclidean distance $AB = \sqrt{(AC)^2 + (CB)^2}$ [8]. In this case, we have squared displacement along the redline and along the green latitude line.

If $p=1$, we will get the Manhattan distance. $AB=AC+CB$ [8]. In this case, we have displacement using the redline and along the green line longitude or latitude but the distance is the sum of the two without squaring.

If $p \rightarrow \infty$, we will get the Chebyshev distance [8] $AB=\text{Max}(|XB - XA|, |YB - YA|)$ the distance is only the bigger displacement of the two directions. In this case, for all the lines longitude and latitude in additions Northeast-southeast and Northwest-southwest.

So: the three distances are summarized in Figure 7.

L _p Minkowski distance measures	
Distance name	Definition
Manhattan	$\sum_{i=1}^n p_i - q_i $
Chebyshev	$\text{Max}_i p_i - q_i $
Euclidean	$\sqrt{\sum_{i=1}^n p_i - q_i ^2}$

Figure 7. Summary of the three distance measures [8]

Let us consider visited locations A, B and D from the user Pasi so that C is a location near to the users Pasi and Radu. It will be interesting to see different values of p for B and D because those two locations are the same as Radu's location but far away to the user Pasi. See figure 8.

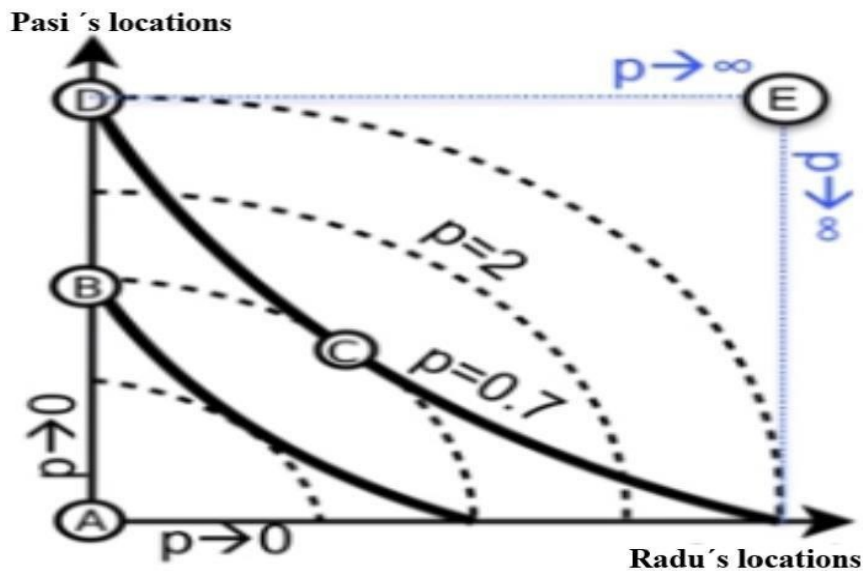


Figure 8. Representation of Minkowski distance between MOPSI users [28]

2.1.2 Bhattacharyya distance

Bhattacharyya distance [8] is used to calculate the similarity of two probability distributions, or the amount of overlap between two statistical samples or populations. Bhattacharyya distance is symmetrical or being invariant to scale in the case of two Gaussian probability density distributions. The mathematical formula of Bhattacharyya distance is:

$$\text{Bhattacharyya distance} = \left| -\ln \sum \sqrt{p_i q_i} \right|$$

We consider 3 users and count the number of visits they did to those points shown in Figure 9 to study similarities between their visit history. The computation of relative frequencies, for example, of the user Julia is done by dividing the number of visits at the location of Science Park (shown with text “Impit” on the map) by the total number of her visits . This results in relative frequency of $1/7 = 0.14$.

Relative frequencies of three users’ visits are shown in Figure 10. Using these statistics, user similarities are shown in Figure 11 with three different distance measures. All measures show that Pekka and Bartek are more similar (bigger distance) than Julia and Bartek.



Figure 9. Count visit statistics [9]









	Frequencies				Relative frequencies		
	Pekka	Bartek	Julia		Pekka	Bartek	Julia
	4	3	1	8	0.44	0.50	0.14
	2	2	0	4	0.22	0.33	0.00
	0	0	3	3	0.00	0.00	0.43
	1	1	1	3	0.11	0.17	0.14
	0	0	2	2	0.00	0.00	0.28
	1	0	0	1	0.11	0.00	0.00
	1	0	0	1	0.11	0.00	0.00
	0	0	0	0	0.00	0.00	0.00
	9	6	7				

Figure 10. Figure 10. Frequencies and relative frequencies of users [9]







Bhattacharyya distance		Manhattan distance		Euclidean distance	
Pekka vs Bartek	Bartek vs Julia				
					
0.47	0.26	0.06	0.36	0.36%	13%
0.27	0.00	0.11	0.33	1.21%	11%
0.00	0.00	0.00	0.43	0.00%	18%
0.14	0.15	0.06	0.03	0.36%	0%
0.00	0.00	0.00	0.28	0.00%	8%
0.00	0.00	0.11	0.00	1.21%	0%
0.00	0.00	0.11	0.00	1.21%	0%
0.00	0.00	0.00	0.00	0.00%	0%
$\Sigma = 0.88$	0.41	$\Sigma = 0.42$	1.43	$\Sigma = 0.04$	0.50
$-\ln = 0.13$	0.89				

Figure 11. Comparison between three different distance measures [9]

2.1.3 Haversine distance calculation

To show how far (in meters) a target is from the user, we use the haversine distance to calculate the great circle distance on the surface of the earth. The points are given via their coordinates: longitude and latitude.

MOPSI application provides many options for the user. Using the distance to services and photos, the distance to the service recommended for the user can be shown. The nearest distance to Pizza Master is 423 meters (see Figure 12). If we recommend data from users' photo collection rather than services, the nearest photo location is at Uuro Shell and is 5 km, as shown in Figure 13.

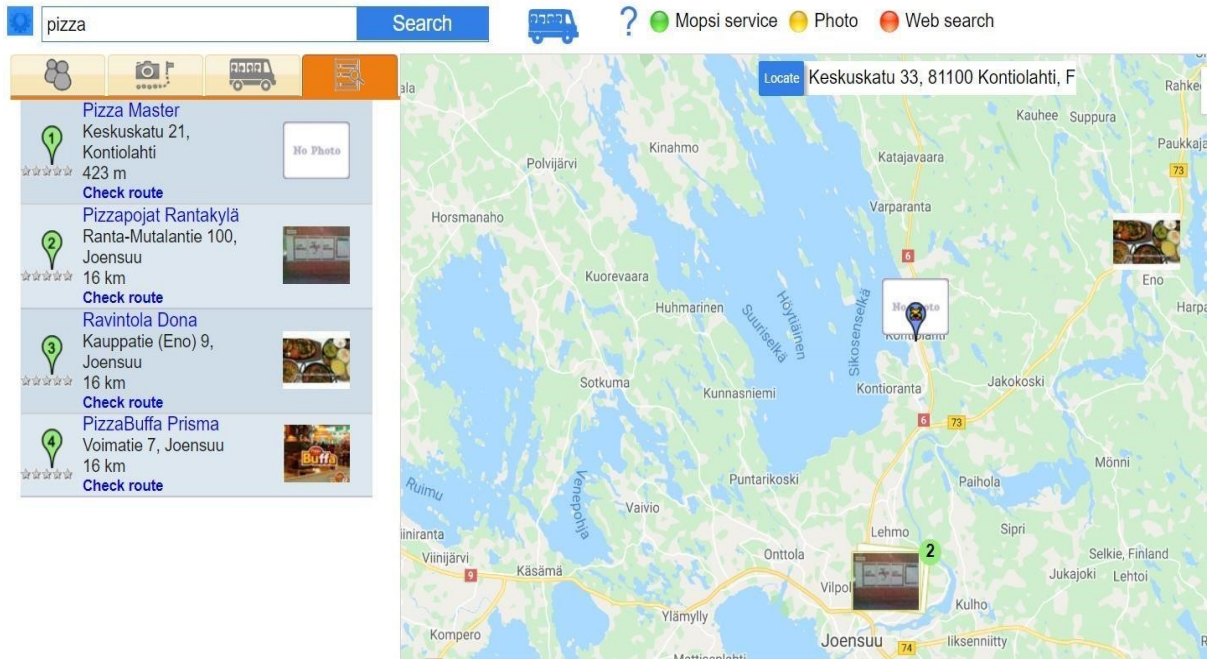


Figure 12. MOPSI options, distance to service [27]

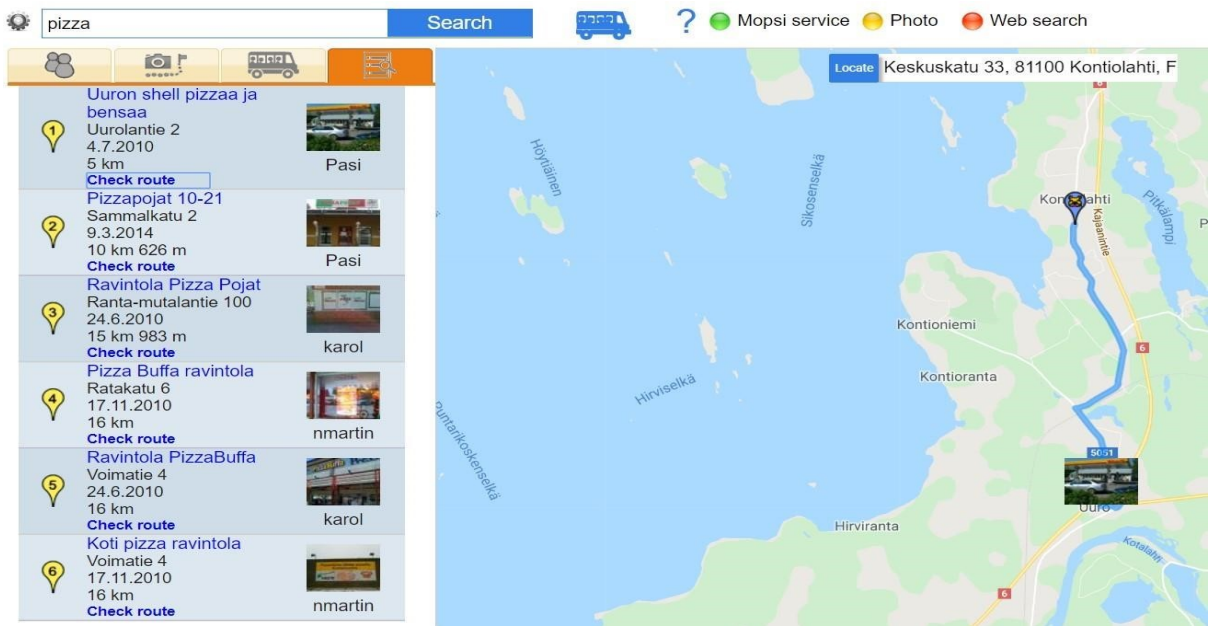


Figure 13. MOPSI options, distance to photos [27]

In the next example, we try to find the distance between Joensuu and Kuopio. Assuming Earth radius is $R = 6371$ km, we define d as the haversine distance between Joensuu and Kuopio where: φ_1, φ_2 are latitudes of Joensuu and Kuopio (in radians), λ_1, λ_2 longitude of Joensuu and Kuopio (in radians). The distance is:

$$d = 2r \arcsin \left(\sqrt{\sin^2 \left(\frac{\varphi_2 - \varphi_1}{2} \right) + \cos(\varphi_1) \cos(\varphi_2) \sin^2 \left(\frac{\lambda_2 - \lambda_1}{2} \right)} \right)$$

So, the distance is $d=139$ km using the Haversine distance between Kuopio and Joensuu where Joensuu city is located at latitude 62.6010° , longitude 29.7636° East of Greenwich and North of the Equator, and Kuopio city is located at latitude 62.8980° , longitude 27.6782° East of Greenwich and North of the Equator.

In general, the factors that impact the travel distance between two cities are the latitude and longitude but also the road network. Another factor that impacts on the distance is transportation: walk, bike, car, train, or even boat. To show a more realistic distance in an urban environment, we selected 4078 routes travelled by three Mopsi users, Pasi, Andrei, and Radu as stored in Mopsi. Knowing the transportation mode, we got the results shown in Figure 14 by comparing the average length, speed, and duration for each movement type (walking, running, biking, skiing, driving).






		Pasi	Andrei	Radu
	Length Speed Duration Proportion	7.8 km 6.6 km/h 1.2 hours 17 %	2.7 km 5.4 km/h 0.6 hours 22 %	3.3 km 4.9 km/h 0.8 hours 44 %
	Length Speed Duration Proportion	13 km 10.8 km/h 1.2 hours 67 %	4.8 km 10.1 km/h 0.5 hours 19 %	4.7 km 10.5 km/h 0.5 hours 15 %
	Length Speed Duration Proportion	12 km 16 km/h 0.9 hours 11 %	13 km 18 km/h 0.7 hours 34 %	25 km 19 km/h 1.2 hours 20 %
	Length Speed Duration Proportion	9.8 km 10.2 km/h 0.9 hours 2 %	11 km 7.7 km/h 1.4 hours 6 %	10 km 7.4 km/h 1.4 hours 6 %
	Length Speed Duration Proportion	20 km 51 km/h 0.6 hours 3 %	74 km 53 km/h 1.3 hours 19 %	54 km 60 km/h 0.9 hours 15 %
No. Routes		2023	626	1429

Figure 14. Distance travelled via tracking in Mopsi [10]

The distance is used as a feature when predicting which destination the user is traveling to [10] given a transportation mode. The transportation mode is important as it impacts the traveled distance. If the user started driving a car, the destination candidate is unlikely to be nearby, whereas if the user is walking, the destination cannot be too far away. Here μ and σ are the mean travel distance and standard deviation. The direction is important when users move optimally towards the destination. The candidates in the opposite direction of the user's movement should receive lower scores.

$$Distance(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

2.2 Travel distances

2.2.1 Road network (OSM / OSRM)

Application Programming Interface (API) are used for software components to communicate with each other. Mopsi uses APIs to implement various functions, see Figure 15

The Open Source Routing Machine (OSRM) is an open-source router designed to be used with road network data. In Mopsi, we use data from Open Street Map (OSM), which is a collaborative project to create a free editable map of the world. The shortest path is calculated using multilevel Dijkstra's algorithm.

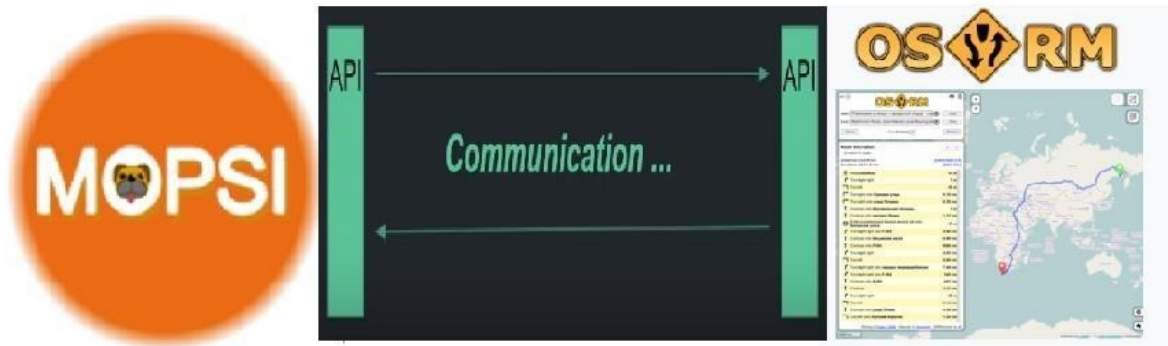


Figure 15. API between MOPSI and OSRM [27]

Overpass API provides query functionality to OSM data. The client sends a query to the API and gets back the data set that is a subset of OSM map data. The selection of the search criteria is like location, type of objects, tag properties, proximity, or combinations of them. It acts as a database backend for various services, see Figure 16.

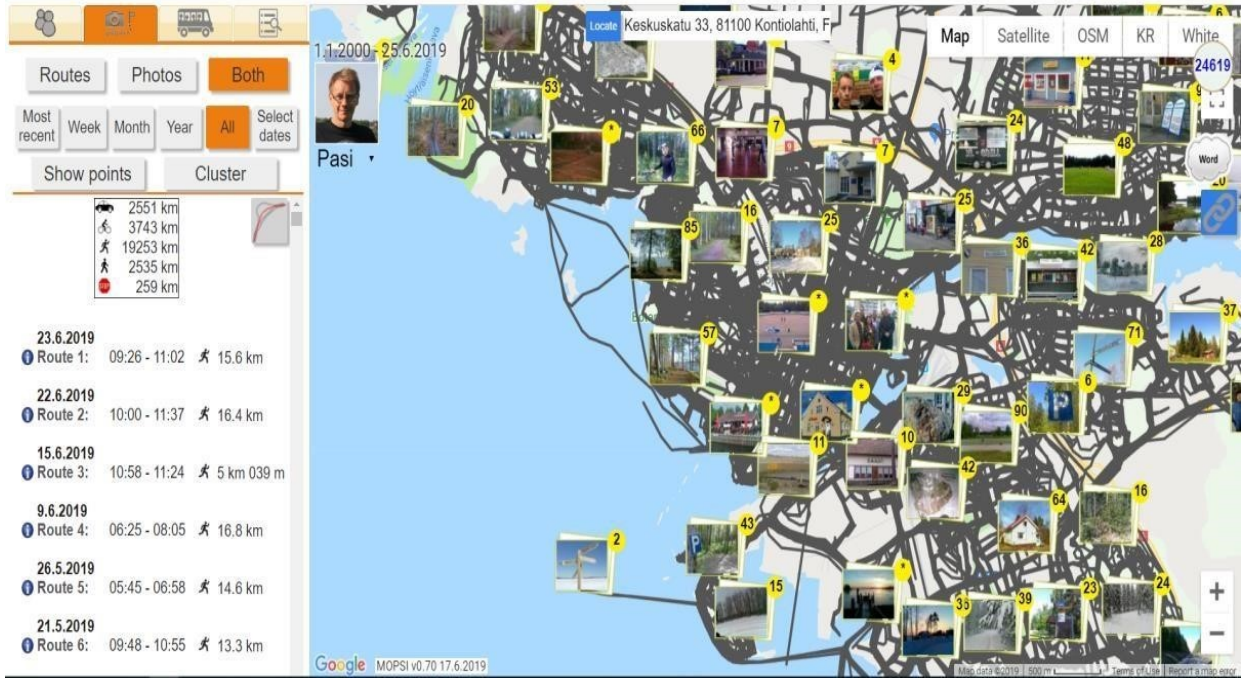


Figure 16. MOPSI's database shown the entire data collection of the user Pasi in Joensuu [27]

2.2.2 Similarity of routes

There are many methods for computing route similarity. They are classified according to their robustness towards noise caused by sensor devices or irregularities in sampling or transcription errors. We can use the Jaccard similarity coefficient and compute the similarity between two routes by dividing the size of their intersection by the size of their union [1].

The method dilates each of the two routes separately for computing the intersections. We denote by C^d the extra cells from a separate set. It is considered as a buffer region when comparing two routes [1].

$$S_{(C_A, C_B)} = \frac{|(C_A \cap C_B) \cup (C_A \cap C_B^d) \cup (C_B \cap C_A^d)|}{|C_A \cup C_B|}$$

$$|C_A \cup C_B| = |C_A| + |C_B| + (C_A \cap C_B)$$

$$J(C_A, C_B) = \frac{|C_A \cap C_B|}{|C_A \cup C_B|}$$

2.2.2.1 C-SIM algorithm

C-SIM algorithm [1] computes the similarity between two given routes in linear time complexity, see Figure 17.

$$O(N_A + N_B + |C_A| + |C_B|)$$

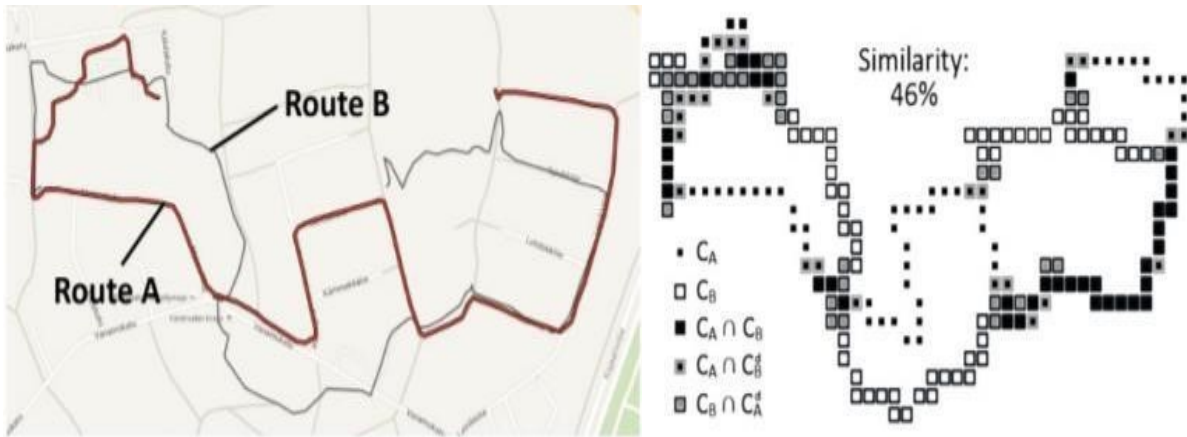


Figure 17. C-SIM algorithm between Route A and B [1]

2.2.2.2 Route Similarity Ranking (RSR) algorithm

The algorithm finds all similar routes in the database for a given input. The algorithm is used in Mopsi to searching among jogging, cycling, and cross-country skiing, and was tested on over 6,700 routes. This service is available for Mopsi users to see their statistics and compare with past attempts or other users that completed similar routes and analyze progress over time. In Figure 18, two routes of user Radu are shown having 70% similarity. One route takes a 1km shorter, off-road path. The other route is quicker, despite the extra kilometer.

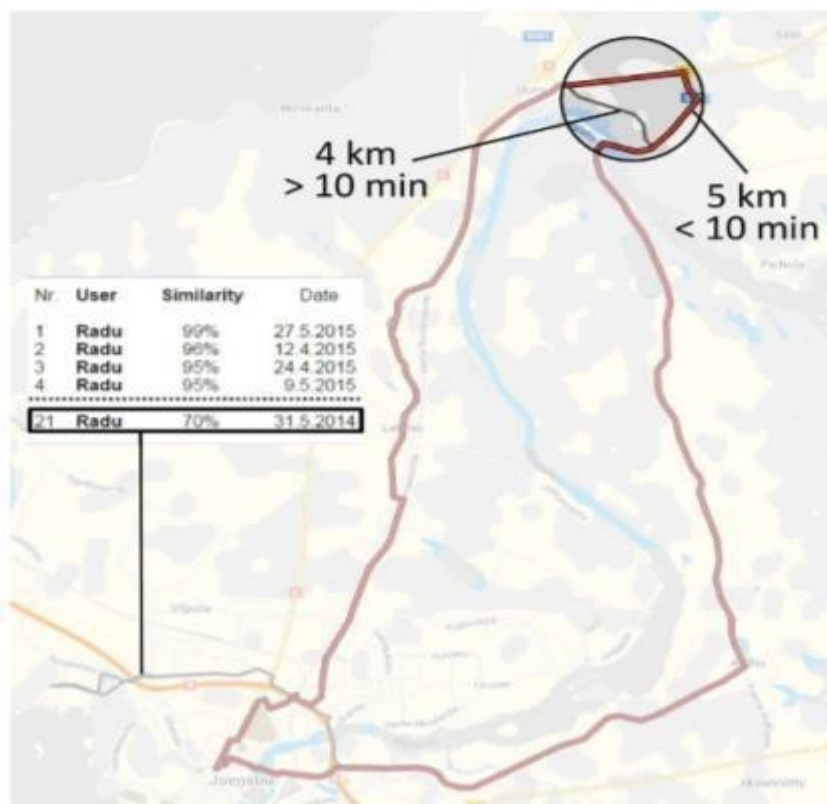


Figure 18. Radu's trajectories similarity [1]

2.2.2.3 Similarity and inclusion

The inclusion measure calculates what proportion of a given route is contained in another. Using the grid, we compute the inclusion between two routes, C_A and C_B , as [1]:

$$I(C_A, C_B) = \frac{|C_A \cap C_B| + |C_A \cap C_B^d|}{|C_A|}$$

Inclusion and similarity have different mathematical formula. The inclusion is not symmetric and rarely gives the same result if we dilate the second route and normalize the result with respect to the original route.

The preliminary version of the inclusion measure was presented in [1]. Algorithm INC has the same time complexity as C-SIM. In Figure 19, 61% of A is included in B and 96% of B is included in A.



Figure 19. Inclusion between Routes A and B [1]

2.2.2.4 Examples

For calculating the similarity, we consider that every trajectory on the Earth is represented as a linked list of cells represented by their center (x =latitude, y =longitude). Radu and Pasi divide the space by generating cells if two consecutively generated cells are not adjacent, we fill the gap by using linear interpolation with the equation:

$$y = \frac{y_2 - y_1}{x_2 - x_1} (x - x_1) + y_1$$

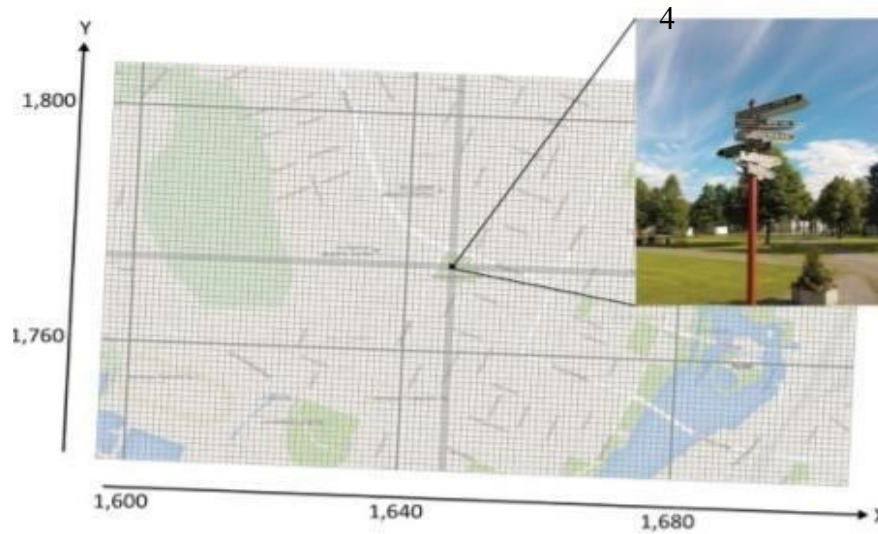


Figure 20. Location defined by latitude and longitude [1]

The highlighted cell in Figure 20 is in the center of a small park in Joensuu. The easting and northing values of the two cells inside a 100km square are 25×25 -meter cells in Joensuu. The procedure of searching most similar routes happens by creating a database that stores many routes. The *familiarity* feature measures if the user has recorded a similar trajectory in the past and if there is a reoccurring pattern [3]. It is used in a system shown in Figure 21, which tries to predict where the user is moving.

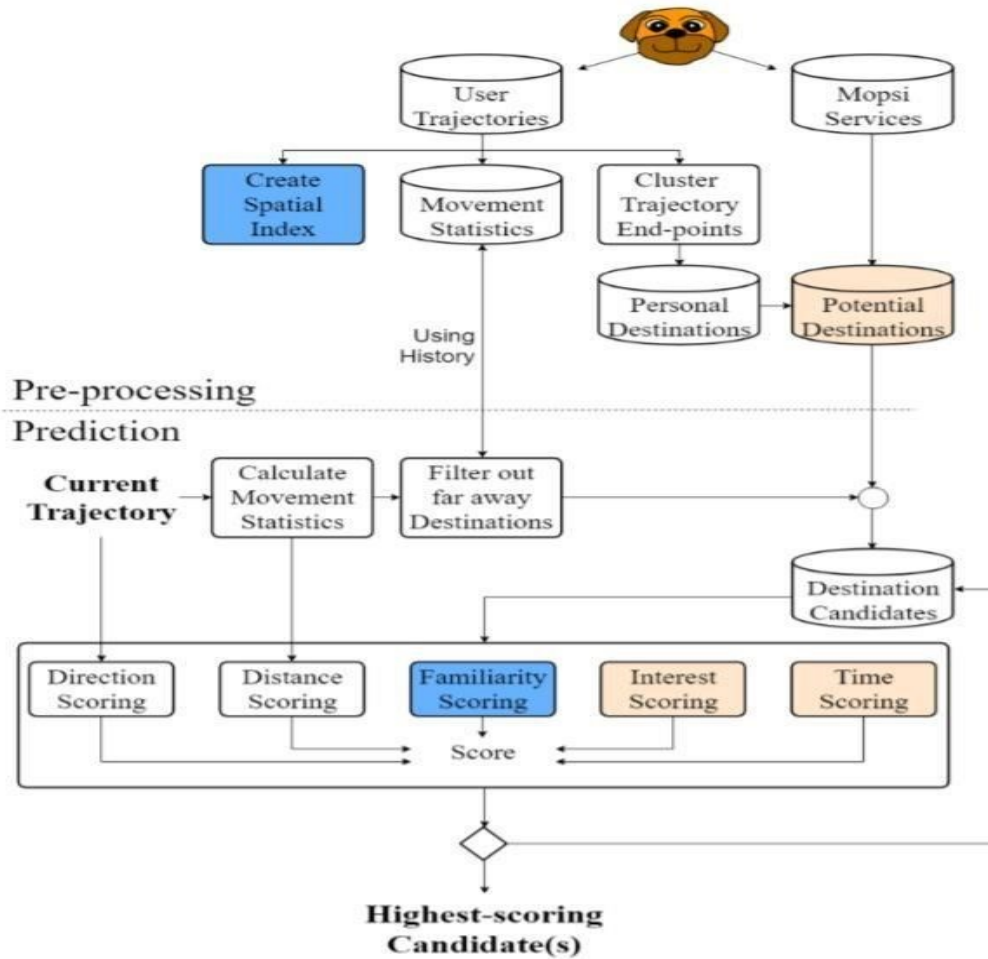


Figure 21. System predicting the destination based on familiar movement patterns [3]

In (Mariescu-Istodor et al, 2019) the overwhelming number of candidates is reduced to the top-ranking candidates by clustering using the inclusion values and keeping only the top cluster. The mathematical equation defines the familiarity feature

$$Familiarity(d) = p(d, S)$$

S is the set of trajectories which include the current trajectory. To obtain S, in the inclusion measure presented (Mariescu-Istodor and Frănti, 2017) the query of the Mopsi trajectory database and rank trajectories happens. An example is given in Figure 22.



Figure 22. Radu's familiar patterns in movement [3]

2.2.3 Tour length in Mopsi

In Mopsi orienteering game called O-Mopsi, the player visits several targets, mainly in a city area. There is no pre-defined order of visiting targets and the game ends immediately after all targets have been reached. Thus, the game playing implicitly includes the open-loop *travelling salesman problem* (TSP). It seems a bit easy to find the best order in the places that connected to each other linearly or near-linearly, as the Otsola example in Figure 23, but is much more difficult in other examples like Hukanhauta and Christmas Star [11].



Figure 23. TSP problem with difficulty level increasing from left to right [11]

So, we can say that TSP has different levels of difficulties related to the form of the network that connects the different places to each other, as well as the number of places visited impacting to problem difficulty to find the shortest path [11]. In closed loop TSP, the goal is to go to all the nodes and to return to the beginning while minimizing the length of the tour. The problem in O-Mopsi is open loop TSP where player do not need to return to the starting node.

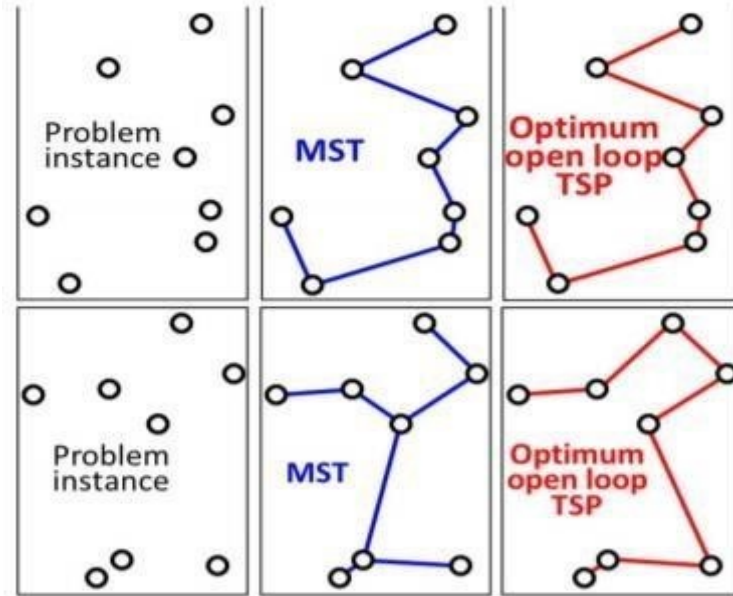


Figure 24. Examples of MST and open loop TSP solutions [11]

It is possible to use the *minimum spanning tree* (MST) to help find the linear or near linear connections that minimize the total distance during the tour [24]. In general, the minimum spanning tree is shorter or equal to the optimum solution for the open-loop TSP because $|MST| \leq |TSP| \leq |SP|$. The solution of the open-loop TSP is also spanning tree but not necessarily the minimum one, see Figure 24.

Kruskal algorithm operates on a spanning forest by greedily merging two spanning trees to extend the entire cost of the spanning forest least. In [24], Kruskal algorithm is modified with the constraint that allows merging two sub-solutions using only by their endpoints. This not only prevents branches to appear within the tree but it also reduces the number of possible links to be considered for the merge. Thus, it enforces the ultimate result to be an answer for the open-loop TSP. It allows potentially more efficient implementation than standard Kruskal for MST. Although this is often still rather far from the optimum, it is useful to provide an affordable estimate for the TSP solution.

The sole limitation we would like to feature is to permit the merges to be made using only the endpoints of the two chains. In the beginning, every node is assigned to its own set. The links are sorted and processed from the shortest to the longest. Every link that connects two distinct sets from their endpoints are merged: thus, avoiding cycles and branches. Merges are continued until only one set remains. The effect of the modification is shown in Figure 25. The positive side is that the number of choices to be considered for the following merge is significantly reduced from standard Kruskal but the negative effect is that the optimality cannot be anymore guaranteed.

The open and closed loops are closely related. However, it is an uphill task to convert the answer from one to the other. For instance, removing the longest link from the closed-loop system solution does not necessarily provide the optimum solution for the open loop case, as proven by counter-example in Figure 26. It is also possible to modify the open loop problem instance to create a closed loop problem instance. One can then solve the problem using algorithm for the closed loop solution. This also indicates that the open loop case is a special case of the closed loop [24].

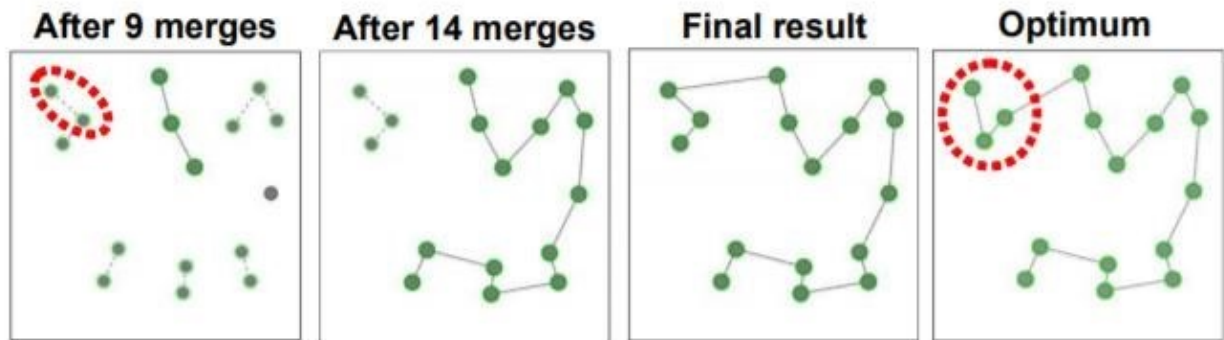


Figure 25. Merging operations [24]

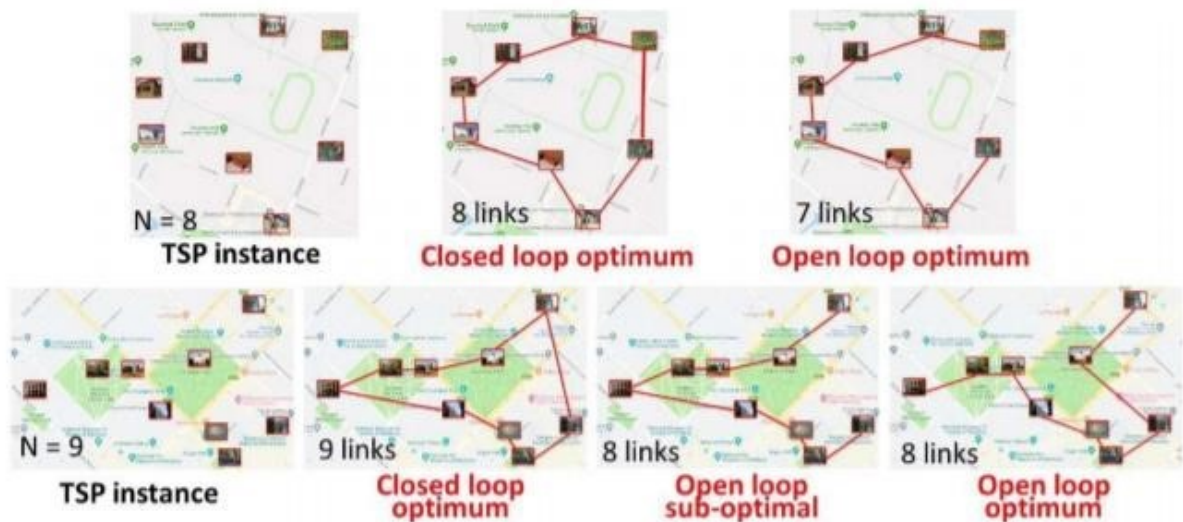


Figure 26. Open and closed loop for travelling salesman problem [24]

3. String distances

String distance is a metric that measures the distance between two text strings. Syntactic measures operate on the words and their characters without any assumption of the language or the meaning of the content and it focuses mainly on specific tasks such as names of people, places, institutions or companies. They point out that the performance of the similarity measures is affected by text length, spelling accuracy, abbreviations, and language [6].

3.1 Distance between two strings

3.1.1 Strings of the same length

Hamming and Lee distances are used in case the strings have the same length. The *Lee distance* between two strings $x_1x_2 \dots x_n$ and $y_1y_2 \dots y_n$ of equal length n over the q -ary $Z_q = \{0, 1, \dots, q-1\}$ of size $q \geq 2$ is a metric defined in [12].

$$\sum_{i=1}^n \min(|x_i - y_i|, q - |x_i - y_i|)$$

If $q > 3$ this is not metric anymore, as the Lee distance can become bigger than 1. For example, when $q = 6$, then the Lee distance between 3140 and 2543 is $1 + 2 + 0 + 3 = 6$. If $q = 2$ or $q = 3$ the Lee distance coincides with the Hamming distance, because both distances are 0 for two single equal symbols and 1 for two single non-equal symbols.

Hamming distance between two strings is the number of mismatches at the same position. It only applies to strings of the same length. For example, the Hamming distance between the strings “point” and “paint” is 1, see Figure 27. Hamming distance can also be defined as the difference between the size of the string and the number of matches.

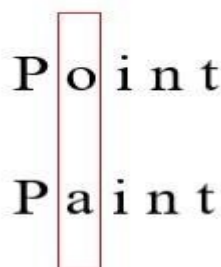


Figure 27. Hamming distance between two strings

3.1.2 Strings of different lengths

The family of distances used for the strings of different lengths, and it still available for the strings of the same length.

3.1.2.1 Levenshtein distance

Levenshtein distance is a string metric for measuring the difference between two sequences and has many applications in strings matching, and in spelling checking. Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) to convert the strings to become equal, see Figure 28. For example, the Levenshtein distance between “SPORT” and “SPORTS” is 1, since the following inserts one into the other by adding character “S”. As well, between “JOENSUU” and “SUU” is 4, since we delete four characters.

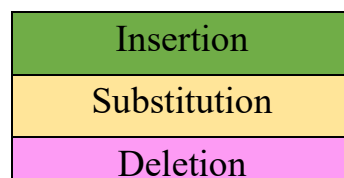


Figure 28. Levenshtein distance operations [6]

The Levenshtein distance is 5 operations to transform the string “ELÄIMISTÖ” to “ELÄMÄ” and the string “ELÄMÄ” to “ELÄIMISTÖ” so Levenshtein distance is symmetric, see Figure 29. It called also *Edit distance* as it calculates the least number of edit operations that are necessary to modify one string to obtain another string, i.e. minimum edit distance.

E	L	Ä	I	M	I	S	T	Ö
E	L	Ä		M	Ä			

Figure 29. Levenshtein distance operations between two strings is 5 consisting of one deletion, three additions and one substitution.

We obtain the Levenshtein distance using dynamic approaches or dynamic programming. We use the Levenshtein matrix that can be filled from the upper left to the lower right corner. Each jump horizontally or vertically corresponds to an insert or a delete. Respectively, the cost is normally set to 1 for each of the operations and we call this method also the diagonal jump can cost either one of the two characters in the row and the column does not match else 0 if they match. Each cell always minimizes the cost locally [13].

The mechanism how to fill Levenshtein matrix is explained in Figure 30 and Figure 31. We obtain the Levenshtein Matrix where the intersection of last column value with the last row value represents the Levenshtein distance described in Figure 32.

Substitution value	Insertion value
Deletion value	Minimum (Substitution, Insertion, Deletion) value +1

Figure 30. Levenshtein distance If column character and row character have different values

Substitution value	Insertion value
Deletion value	Substitution value

Figure 31. Levenshtein distance If column character and row character have equal values

		E	L	Ä	I	M	I	S	T	Ö
	0	1	2	3	4	5	6	7	8	9
E	1	0	1	2	3	4	5	6	7	8
L	2	1	0	1	2	3	4	5	6	7
Ä	3	2	1	0	1	2	3	4	5	6
M	4	3	2	1	1	2	3	4	5	6
Ä	5	4	3	2	3	2	3	4	5	6

Figure 32. Levenshtein Matrix.

3.1.2.2 Damerau–Levenshtein distance

Damerau–Levenshtein distance is a string metric for measuring the edit distance between two sequences. It uses all the operations used in the Levenshtein distance like substitution, insertion, deletion. In addition, the transposition of two adjacent characters that require the change one word into the other. for example, "MARTHA", and "MARHTA" where the transposition happens between "T" and "H". [14]

3.2 Distances between sets of strings

3.2.1 Longest common subsequence problem

Longest common subsequence (LCS) problem is the problem of finding the longest subsequence common to all sequences in a set of sequences (often just two sequences). For example, consider the sequences in Figure 33. So, “c d g i” is the longest common subsequence between sequence 1 and sequence 2. So, “c d g i” is the longest common subsequence between sequence 1 and sequence 2.

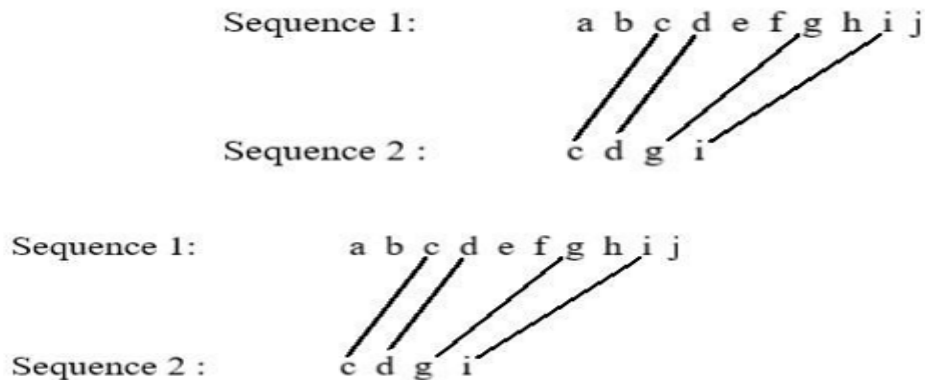


Figure 33. Longest common subsequence between two sequences

3.3.2 String segmentation

Two approaches exist to segment the string: *q-grams* and *tokenization* [6]. Examples of them are shown in Figure 34. The first approach *q-grams* divides a string s into substrings of length q and obtains a sliding window of length q over the string to consider also substrings of length $q - 1$ and to recognize prefixes and suffixes of the string. So-called padding characters ($\# \% \$$) are appended to the beginning and end of the string. The similarity is calculated as follows:

$$QGramsSim(s_1, s_2) = 1 - \frac{\sum_{i=1}^n |match(q_i, Q_{s1}) - match(q_i, Q_{s2})|}{|Q_{s1}| + |Q_{s2}|}$$

Where Q_{s1} and Q_{s2} are the multi-sets of q -grams from s_1 and s_2 , respectively, $n = |Q_{s1} \cup Q_{s2}|$, and $match(q_i, Q_{s1})$ is the number of times the q -gram q_i appears in Q_{s1} . In [6], q -grams with paddings is used to consider tokens having fewer than q characters, such as the determiners “a” and “an”.

Tokenization divides a string into units called token using whitespaces and punctuation characters. The rationale behind tokenization is to utilize the data at the token level and to beat problems of token swap and missing tokens. There are two solutions to unravel the token ordering issue were introduced: *sorting heuristic* and *permuting heuristic* [6].

In sorting heuristic, each string is tokenized, tokens alphabetically ordered, rejoined again, then edit distance is applied to the modified strings. In permuting heuristic, all token permutations are obtained from the primary string and a comparison between all the permuted strings and the second string is performed; the best similarity value is chosen. However, these heuristic solutions are inefficient for other sorts of mismatching like missing tokens, especially when the length of the absent token is taken into account like “Rosso” and “Rosso restaurant”. a much better solution is therefore needed in such cases.

Segmentation method	Output
None (character sequence)	the club at the ivy
q-grams ($q=3$)	the, he_, e_c, _cl, clu, lub, ub_, b_a, _at, at_, t_t, _th, the, he_, e_i, _iv, ivy
q-grams with padding characters	##t, #th, the, he_, e_c, _cl, clu, lub, ub_, b_a, _at, at_, t_t, _th, the, he_, e_i, _iv, ivy, vy%, y%%
1-skip-grams	t'e, h'c, e'l, c'u, l'b, u'a, b't, a't, t'h, t'e, h'i, e'v, l'y
Tokenization	the, club, at, the, ivy

Figure34. The segmentation of String” The club at the Ivy” [6]

3.3.3 Bag-of-tokens

Bag of tokens or words is used for information retrieval from texts in [6]. Here, a text (such as a sentence or a document) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. The selection of suitable matching techniques (set, sequences, bag of tokens) impacts the result of similarity more than the string distances. An example is given in Figure 35.

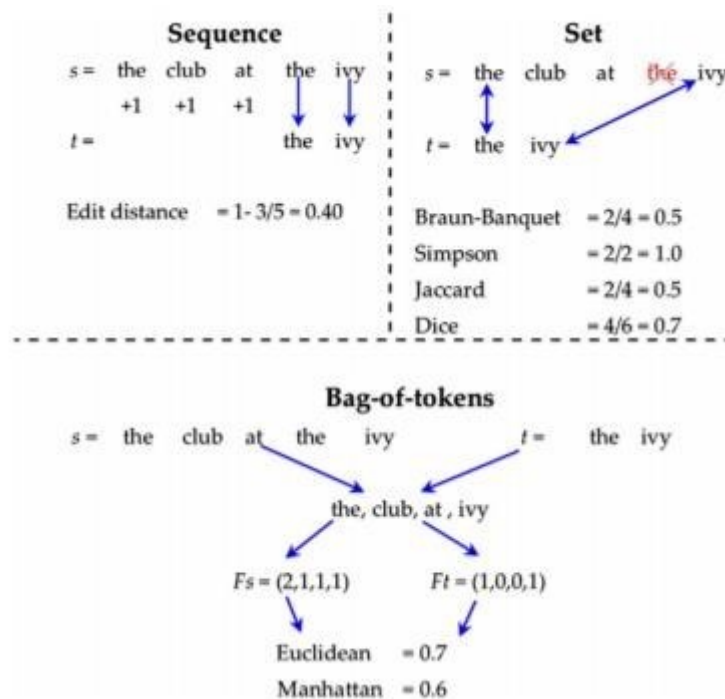


Figure 35. Strings matching techniques [6]

The solution with Set finds the matching using any set-matching method. It can then be applied to measure the overlap between the sets using like Braun-Banquet, Simpson coefficient, Jaccard index, and Dice coefficient, which divide the cardinality of the intersection by the cardinality of the largest set and the smallest set [6]. Sequence matching uses the principle of the operations like insertion, deletion, and substitution at the word level.

4. Other distances

4.1 Multidimensional distances

A space-filling curve is a way of mapping a discrete multidimensional space into one dimensional space. It imposes a linear order for points in multidimensional space [7]. Hilbert curve [15] is a useful curve because it gives a mapping between 1D and 2D space that preserves locality well [8]. In O-Mopsi, user movement provides trajectory that fills the space but the difference is that space-filling curves fill every single position in space but O-Mopsi user only the part needed to find the goals. When reaching each goal, the game plays a sound that is represented by a frequency. The closer the player is to the next goal, the higher the frequency value played as shown in Figure 36. Examples of space-filling curves are shown in Figures 37 and 38.



Figure 36. Distance and sound frequency used in O-Mopsi [5]

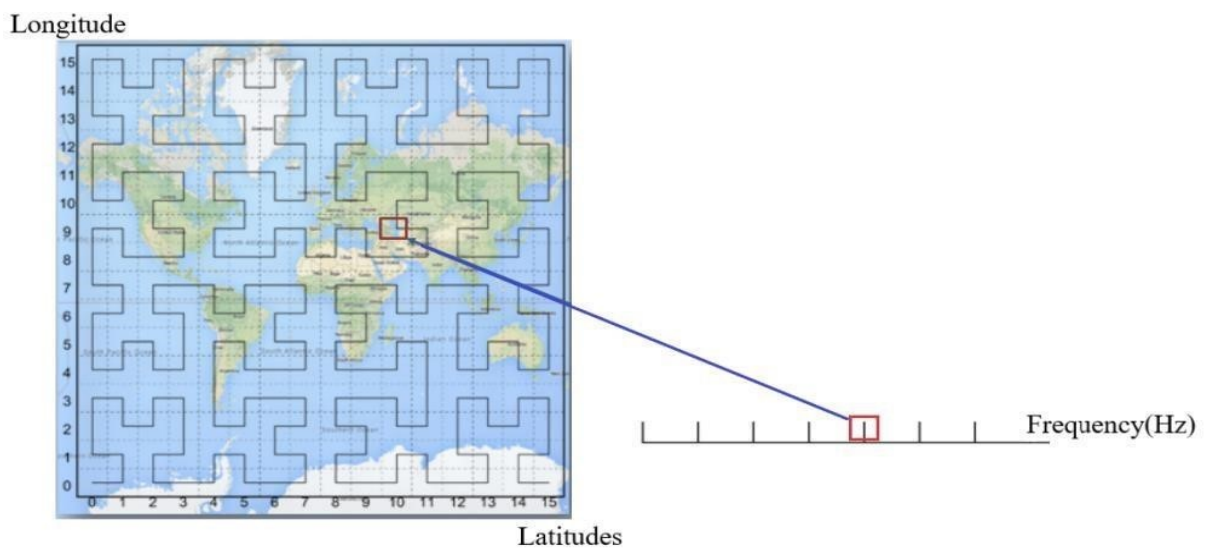


Figure37. Hilbert curve for covering the entire planet in 2 D projection.

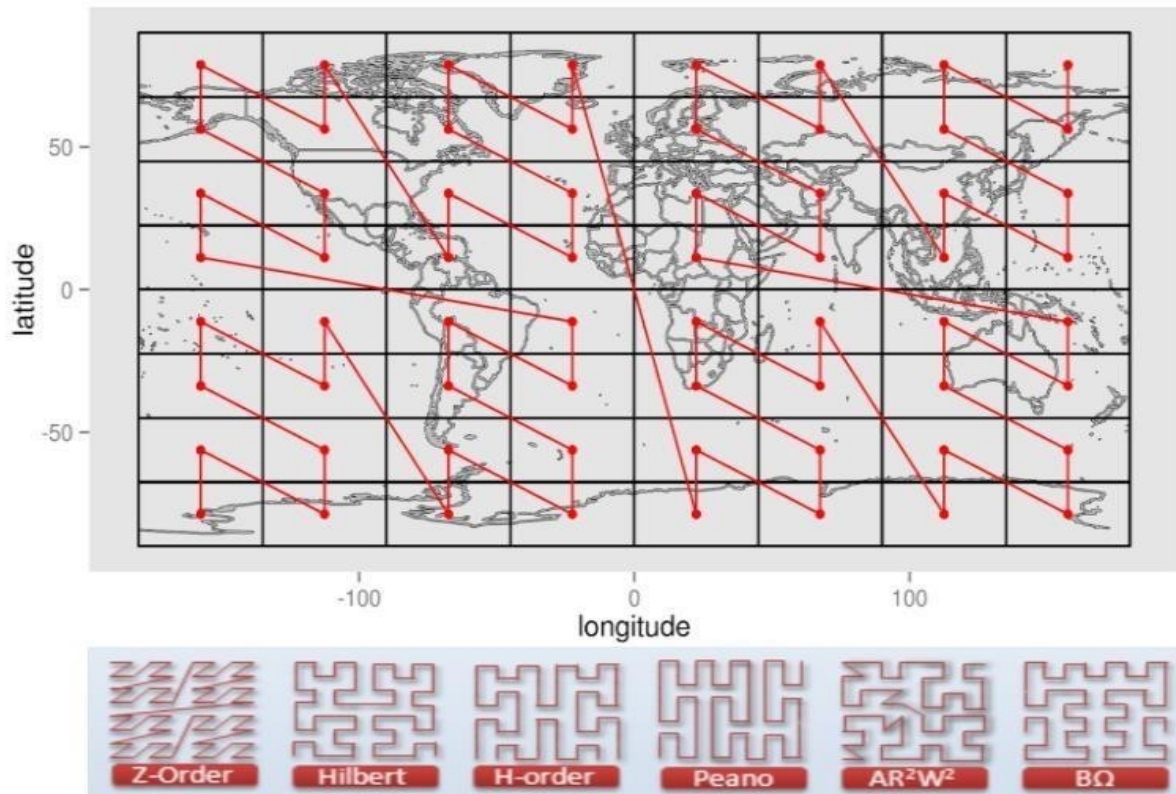


Figure 38. world map in Z-order curve and other curves [16]

4.2 Pairwise distance (Alignment)

Pairwise distance measured between each pair of variables leaves out some of the information in the data matrix M , reducing it to a simple table of pairwise distances. However, it seems that in many cases most of the evolutionary information is conveyed in these distances [17].

4.2.1 Dynamic time wrapping

Dynamic time warping (DTW) is used for measuring the similarity between two temporal sequences, which may vary in speed. For instance, similarities in Radu's activities could be detected using DTW based on the shape of different routes even if there were accelerations and decelerations during the process of observation.

We consider the foremost interesting activities of the user Radu that is mostly walking, cycling, and skiing, see Figure 39. The activities show the speed and distance traveled by the user Radu. There are several patterns. On the highest right, there is a long-distance cycling cluster with varying speeds. The shorter distance (40 km) corresponds to typical training and the longer (60–70 km) corresponds to long runs, which appear less often. The skiing cluster contains many observations with about 15–20 km long including few spurious long-distance ones.

The bottom clusters are commuting by walking and bicycle. Let us now consider two events: one is 55-km cycling at 23 km/h, and another 35 km skiing at 14 km/h. Moving activities of user Radu plotted as a function of the typical speed and total distance (left). His most common activities are skiing, bicycle, and walking, which is emphasized using borders. Two forthcoming events are then shown as red and blue circles with their target speed and distance (right) from Radu's activity data. The forthcoming cycling event is suitable for Radu, but the skiing event is not because its speed is higher than what Radu is capable of [18].

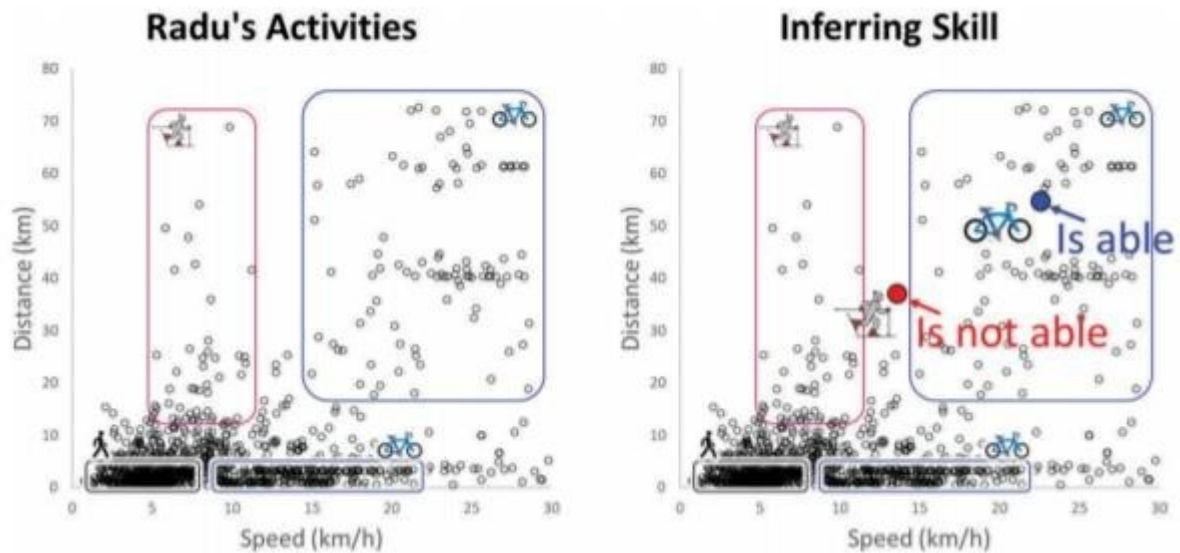


Figure 39. Radu's Activities [18]

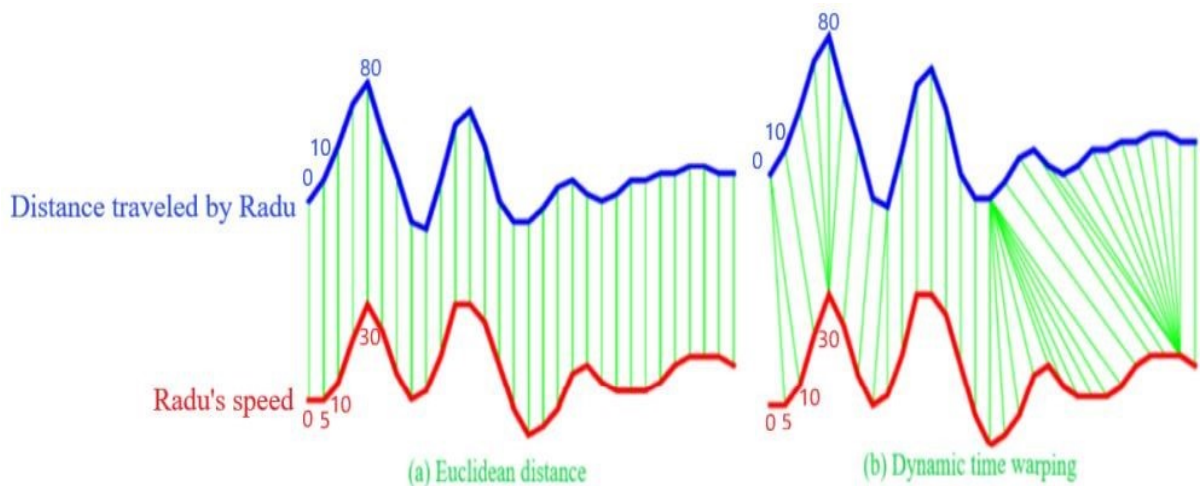


Figure 40. Comparison between Euclidean distance and DTW [19].

4.2.2 Matrix distance and construction of the warping path

The distance matrix is a square matrix (two-dimensional array) containing the distances, taken pairwise, between the elements of a set, which in this case contains the distances travelled by Radu and his speed. Depending on the application, the distance used to define this matrix may or may not be a metric. We construct the warping path using backtracking and greedy search is constructed to minimize the distance between the speed and length. We start from the intersection of the biggest value of the distance travelled by Radu and his speed. After, we select the minimum distance from the three values adjacent of distances. We consider, the pair of sequences are distance and speed for finding the warping path selected by green color on the matrix described in Figure 41.

This is the principle of *Dynamic Time Warping* (DTW) that finds a mapping between positions and minimizes the total distance. Using, the equation of DTW that defined recursively by:

$$DTW_{Distances_X Speed_s}(i,j) = \text{Distance}(X^{(i)}, S^{(j)}) + \min \begin{cases} DTW_{Distances_X Speed_s}(i, j - 1) \\ DTW_{Distances_X Speed_s}(i - 1, j) \\ DTW_{Distances_X Speed_s}(i - 1, j - 1) \end{cases}$$

$Distances_X = \langle X_{(1)}, X_{(2)}, X_{(3)}, \dots, X_{(i)} \rangle$ and $Speed_s = \langle S_{(1)}, S_{(2)}, S_{(3)}, \dots, S_{(j)} \rangle$.

80	320	285	250	220	210	165
70	245	215	185	160	155	115
60	180	155	130	110	140	75
50	125	105	85	70	55	45
40	80	65	50	40	30	25
30	45	35	25	20	15	15
20	20	15	10	10	15	25
10	5	5	10	20	35	55
0	5	10	15	20	25	30

Figure 41. Matrix distance with the wrapping path.

4.3 Multiple Tours Alignment and Tours similarity

Multiple tours alignment is basically an alignment of more than 2 tours, and finding the similarity among the multiple tours. As the model of multiple tour alignment we will consider it computes the pairwise distances in first after it uses order-based evolutionary algorithm [20].

The objective of single traveling salesman is to find the shortest path that visits all cities (26 cities) in a partial map of Finland map showed in Figure 42, given a collection of cities and the cost of travel between each pair. This problem has the following restrictions. The salesman must visit each city one time and must return to the starting city. We consider a tour that connects all the cities.

Given a set of k Tours $T = \{t_1, \dots, t_k\}$ represents a salesman tour, and a multiple tour alignment $A = \{a_1, \dots, a_k\}$ consists in inserting gaps “-” into the original strings of S to make them all the same length. We define the score of a pair denoted as $score(a_i[m], a_j[m])$ where a_i and a_j be a pair of tours from a multiple tour alignment A and defined for each column m as:

$$score(a_j[m], a_i[m]) = \begin{cases} c_{match} & a_j[m] = a_i[m] \\ c_{mismatch} & a_j[m] \neq a_i[m] \\ c_{null} & a_j[m] = a_i[m] = "0" \text{ (in smith waterman algorithm only)} \\ c_{gapped} & a_j[m] = "-" \text{ or } a_i[m] = "-" \end{cases} \quad (\text{Equation 1})$$

We consider, C_x usually takes the following values: $C_{match}=+5$, $C_{mismatch}=-1$, $C_{gapped}=-2$, $C_{null}=0$.

The total score of the pair $score(a_i, a_j)$ is the cumulative score for all columns:

$$score(a_j, a_i) = \sum_{l=1}^k score(a_j[l], a_i[l])$$

To model the MTA problem such as a TSP problem, a pair of tours (t_i, t_j) is seen as two cities (c_i, c_j) and the distance between each pair of cities is associated to the score of the aligned pair (a_i, a_j) . Scores for all aligned pairs form a triangular matrix $T = \{t_{ij}\}$ and using T , a distance matrix $D = \{d_{ij}\}$, with $d_{ij} > 0, \forall i, j$ where t_{max} is the maximum score in matrix T , is computed by:

$$d_{ij} = t_{max} - t_{ij} + 1$$

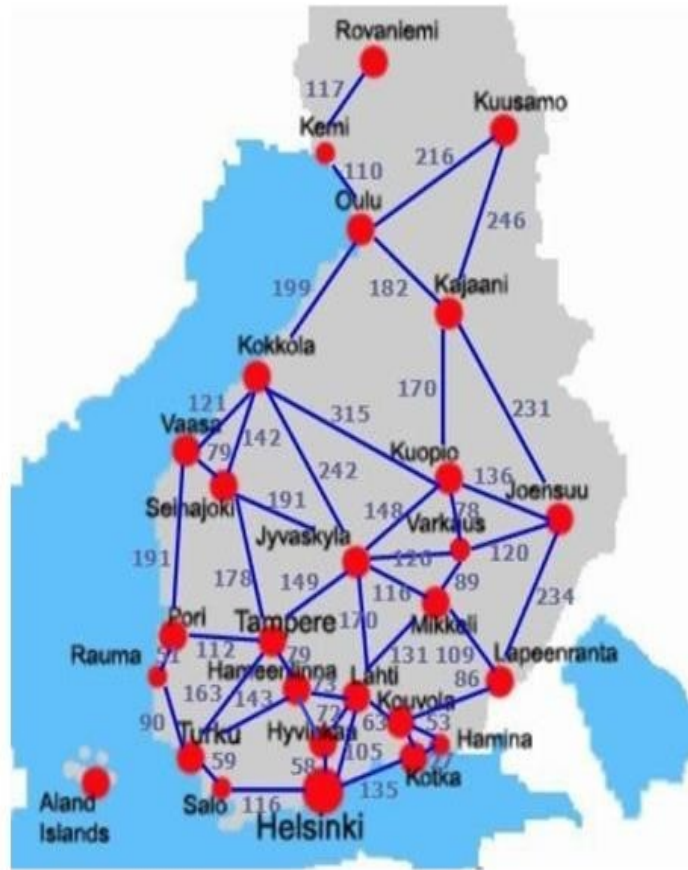


Figure 42. Graph on a Finland Map [25]

All pairs (t_i, t_j) are aligned using the Needleman-Wunsch algorithm and build the matrix T using the scores. In the considered case, we will take two tours of salesman as shown in Figure 43. The first sequence (tour) represented by green color and the second sequence represented by yellow color in a partial map of Finland. The tours are of the same length so we will not need to use gap operation for computing the score Matrix described in Figure 44.

Needleman–Wunsch algorithm is used to find the best alignment of a two salesman tours using the Substitution matrix. We fill the substitution matrix using the matching, mismatching, and gapping coefficients and the value of the next cell $C(i,j)$ by comparing the values of $C(i-1,j-1)$, $C(i-1,j)$ and $C(i,j-1)$ and selecting the maximum value of those cell and copied for $C(i,j)$ as shown in Figure 44.

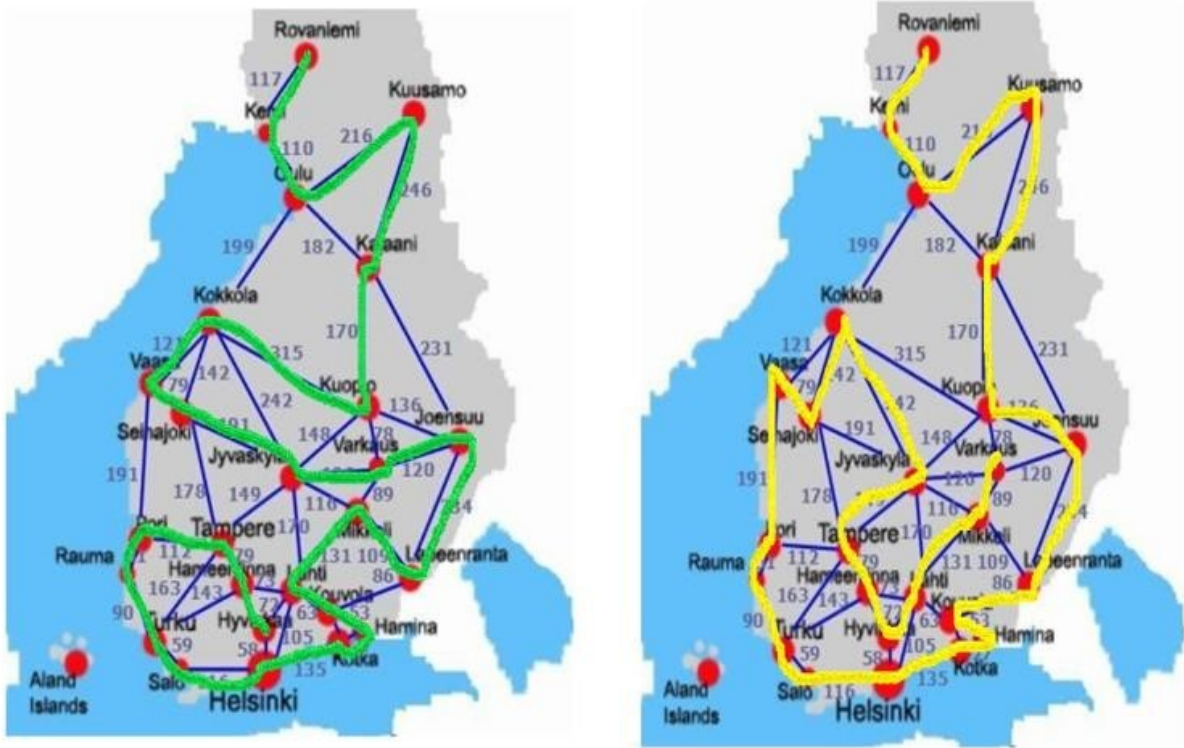


Figure 43. Two tours of a salesman [25]

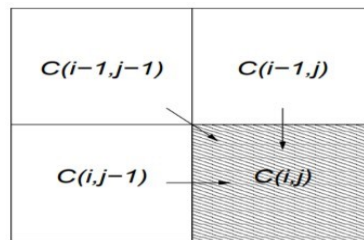


Figure 44. Substitution matrix

If the tour i matches with tour j then:

$$C(i, j) = \text{Max}[(C(i, j-1), C(i-1, j)) + c_{gapped}, C(i-1, j-1) + c_{match}]$$

If the tour i mismatches with tour j , then:

$$C(i, j) = \text{Max} [(C(i, j-1), C(i-1, j)) + c_{gapped}, C(i-1, j-1) + c_{mismatch}]$$

The substitution matrix t_{ij} is used for tour similarities showed a different scores values for Needleman Wunch algorithm based on the global alignments and smith Waterman based on local alignments [20]

		Rovaniemi	kemi	Oulu	kuusamo	kajaani	Kuopio	Joensuu	Lappeenranta	Kouvola	Hamina	kotka	Helsinki	Salo	Turku	Rauma	Pori	Vaasa	Seinäjoki	Kokkola	Jyväskylä	Tampere	Hämeenlinna	Hyvinkää	Lahti	Mikkeli	Varkaus
	0	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22	-24	-26	-28	-30	-32	-34	-36	-38	-40	-42	-44	-46	-48	-50	-52
Rovaniemi	-2	5	3	1	-1	-3	-5	-7	-9	-11	-13	-15	-17	-19	-21	-23	-25	-27	-29	-31	-33	-35	-37	-39	-41	-43	-45
kemi	-4	3	4	2	0	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22	-24	-26	-28	-30	-32	-34	-36	-38	-40	-42	-44
Oulu	-6	1	2	9	7	5	3	1	-1	-3	-5	-7	-9	-11	-13	-15	-17	-19	-21	-23	-25	-27	-29	-31	-33	-35	-37
kuusamo	-8	-1	0	7	14	12	10	8	6	4	2	0	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22	-24	-26	-28	-30
kaajani	-10	-3	-2	5	12	19	17	15	13	11	9	7	5	3	1	-1	-3	-5	-7	-9	-11	-13	-15	-17	-19	-21	-23
Kuopio	-12	-5	-4	3	10	17	18	16	14	12	10	8	6	4	2	0	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22
Kokkola	-14	-7	-6	1	8	15	16	17	15	13	11	9	7	5	3	1	-1	-3	-5	-1	-3	-5	-7	-9	-11	-13	-15
Vaasa	-16	-5	-3	-1	6	13	14	15	16	14	12	10	8	6	4	2	0	4	2	0	-2	-4	-6	-8	-10	-12	-14
Seinäjoki	-18	-7	-5	-3	4	11	12	13	14	15	13	11	9	7	5	3	1	-1	9	7	5	3	1	-1	-3	-5	-7
Jyväskylä	-20	-9	-11	-13	2	9	10	11	12	13	11	9	10	8	6	4	2	0	-2	-4	11	9	7	5	3	1	-1
Varkaus	-22	-7	-9	-11	0	7	8	9	10	11	12	10	8	9	7	5	3	1	-1	-3	-5	-7	-9	-11	-13	-15	6
Joensuu	-24	-9	-11	-13	-2	5	6	13	11	9	7	5	3	1	-1	-3	-5	-7	-9	-11	-13	-15	-17	-19	-21	-23	-25
Lappeenranta	-26	-7	-9	-11	-4	3	4	5	18	16	14	12	10	8	6	4	2	0	-2	-4	-6	-8	-10	-12	-14	-16	-18
Mikkeli	-28	-5	-7	-9	-2	1	2	3	4	5	6	7	9	9	7	5	3	1	-1	0	-2	-4	-6	-8	-10	-9	-7
Lahti	-30	-3	-9	-8	-4	-1	0	1	2	3	4	5	7	8	8	6	4	2	0	-2	-2	-4	-6	-8	-3	-2	-4
kouvola	-32	-5	-4	-6	-6	-3	-2	-1	0	7	8	9	10	12	13	13	11	9	7	5	3	1	-1	-3	-5	-7	-9
Hamina	-34	-3	-5	-5	-7	-5	-4	-3	-2	5	12	13	14	15	17	18	18	16	14	12	10	8	6	4	2	0	-2
kotka	-36	-1	-3	-5	-7	-7	-6	-5	-4	3	10	17	15	13	14	16	17	17	15	13	11	9	7	5	3	1	-1
Helsinki	-38	-3	-2	-1	-3	-5	-7	-7	-6	1	8	15	22	20	18	16	15	15	16	14	12	10	8	6	4	2	0
Salo	-40	-5	-4	-3	-2	-4	-6	-8	-8	-1	6	13	19	28	26	24	22	20	18	16	14	12	10	8	6	4	2
Turku	-42	-7	-6	-5	-4	-3	-5	-2	0	-3	4	11	17	26	33	31	29	27	25	23	21	19	17	15	13	11	9
Rauma	-44	-9	-8	-7	-6	-1	-3	-4	-2	-1	2	9	15	24	31	38	36	34	32	30	28	26	24	22	20	18	16
Pori	-46	-11	-10	-9	-8	-3	-2	-4	-4	-3	0	7	13	22	29	36	43	41	39	37	35	33	31	29	27	25	23
Tampere	-48	-13	-12	-11	-10	-5	-4	-3	-5	-5	-2	5	11	20	27	34	41	42	40	38	36	40	29	30	28	26	24
Hämeenlinna	-50	-15	-14	-13	-12	-7	-6	-5	-4	-6	-8	3	9	18	25	32	39	40	41	39	37	35	45	43	41	39	37
Hyvinkää	-52	-13	-16	-15	-14	-9	-8	-7	-6	-8	-10	1	7	16	23	30	37	38	39	40	38	36	43	50	48	46	44

Figure 45. Substitution matrix for travelling salesman problem using Needleman Wunsch Algorithm

		Rovaniemi	kemi	Oulu	kuusamo	kajaani	Kuopio	Joensuu	Lappeenranta	Kouvola	Hamina	kotka	Helsinki	Salo	Turku	Rauma	Pori	Vaasa	Seinäjäki	Kokkola	Jyväskylä	Tampere	Hämeenlinna	Hyvinkää	Lahti	Mikkeli	Varkaus
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rovaniemi	0	5	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
kemi	0	3	10	8	6	4	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Oulu	0	1	8	15	13	11	9	7	5	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
kuusamo	0	0	6	13	20	18	16	14	12	10	8	6	4	2	0	0	0	0	0	0	0	0	0	0	0	0	0
kaajani	0	0	4	11	18	25	23	21	19	17	15	13	11	9	7	5	3	1	0	0	0	0	0	0	0	0	0
Kuopio	0	0	2	9	16	23	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2	0	0	0	0	0	0
Kokkola	0	0	0	7	14	21	28	29	27	25	24	22	20	18	16	14	12	10	8	6	4	2	0	0	0	0	0
Vaasa	0	0	0	5	12	19	26	27	28	26	24	23	21	19	17	15	13	17	15	13	11	9	7	5	3	1	0
Seinäjäki	0	0	0	3	10	17	24	25	26	27	25	23	22	20	18	16	14	15	22	20	18	16	14	12	10	8	6
Jyväskylä	0	0	0	1	8	15	22	23	24	25	26	24	22	20	18	16	14	13	20	21	25	23	21	19	17	15	13
Varkaus	0	0	0	0	6	13	20	21	22	23	24	22	23	21	19	17	15	13	18	19	23	24	22	20	18	16	20
Joensuu	0	0	0	0	4	11	18	25	23	21	22	23	21	22	20	18	16	14	16	17	21	22	20	21	19	17	18
Lappeenranta	0	0	0	0	2	9	16	23	30	28	26	24	22	20	21	19	17	15	14	15	19	20	21	19	20	18	16
Mikkeli	0	0	0	0	0	7	14	21	28	29	27	25	23	21	19	20	18	16	14	13	17	18	19	20	18	25	23
Lahti	0	0	0	0	0	5	12	19	26	27	28	26	24	22	20	18	19	17	15	13	15	16	17	18	25	23	21
kouvola	0	0	0	0	0	3	10	17	24	31	29	27	25	23	21	19	17	18	16	14	12	14	15	16	23	24	22
Hamina	0	0	0	0	0	1	8	15	22	29	36	34	32	30	28	26	24	22	20	18	16	14	13	14	21	22	23
kotka	0	0	0	0	0	0	6	13	20	27	34	41	39	37	35	33	31	29	27	25	23	21	19	17	15	13	11
Helsinki	0	0	0	0	0	0	4	11	18	25	32	39	46	44	42	40	38	36	34	32	30	28	26	24	22	20	18
Salo	0	0	0	0	0	0	2	9	16	23	30	37	44	51	49	47	45	43	41	39	37	35	33	31	29	27	25
Turku	0	0	0	0	0	0	0	7	14	21	28	35	42	49	56	54	52	50	48	46	44	42	40	38	36	34	32
Rauma	0	0	0	0	0	0	0	5	12	19	26	33	40	47	54	61	59	57	55	53	51	49	47	45	43	41	39
Pori	0	0	0	0	0	0	0	3	10	17	24	31	39	45	52	59	66	64	62	60	58	56	54	52	50	48	46
Tampere	0	0	0	0	0	0	0	1	8	15	22	29	37	43	50	57	64	62	60	58	59	63	61	59	57	55	53
Hämeenlinna	0	0	0	0	0	0	0	0	6	13	20	27	35	41	48	55	62	63	61	59	57	61	68	66	64	62	60
Hyvinkää	0	0	0	0	0	0	0	0	4	11	18	25	33	39	46	53	60	61	62	60	58	59	66	73	71	69	67

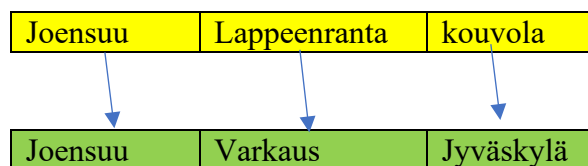
Figure 46. Substitution matrix for travelling salesman problem using Smith and waterman Algorithm

From Figure 45, we conclude that the scores value $t_{max}=44$ corresponds to the intersection of the last cell of the tour i with the last cell tour j using Needleman Wunsch algorithm. In Needleman Wunsch algorithm the first row and column are subject to gap penalty and the score can be negative and we consider the traceback from the beginning of the cell at the lower right of the matrix to the ending at the top left of the cell

Needleman Wunsch algorithm also uses the concept of the global alignment between the two tours travelled by the salesman and finds the similarity between the two tours. However, in Smith Waterman algorithm the value 0 is set for the first column and row and as shown in (Equation 1) during the initialization. The negative score is set to 0 and we identify the traceback using from the beginning with the highest score and ending when 0 is encountered.

From Figure 46, we conclude that the scores value $t_{max}=73$ corresponds to the highest score in Smith Waterman algorithm. By following the traceback we found that there are 5 similar local alignments. We will consider only two local alignment:

- First local alignment



- Second local alignment

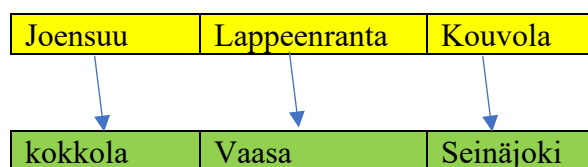


Figure 47. local alignments similarities

Another algorithm called BLAST (basic local alignment search tool) is used for finding how similarity the salesman tours have by searching and comparing with a query tour with those in the databases [22]. For finding the best query tour we will try to discover the longest common way travelled by the salesman during the tours.

While BLAST is faster than any Smith-Waterman implementation for most cases, it cannot "guarantee the optimal alignments of the query and database sequences" as the Smith-Waterman algorithm does according to [22]. The optimality of Smith-Waterman also "ensured the best performance on accuracy and the most precise results" at the expense of time and computer power [22].

We will try to find the longest common tour followed by salesman during his tours for finding a suitable query tour that we will compare it with the other tours. The both tours are matching in the cities:

Rovaniemi -Kemi- Oulu-Kuusamo- Kajaani-Kuopio

We symbolize to the matching cities of sub tour by k -mers algorithm [22]. It grows exponentially with the value of k [22]. The idea of the algorithm is to find the salesman tours similarity by looking for database similarity and searching between the tours by scanning the most matching sub tours [26]. The algorithm pseudo code is below.

```
1: PROCEDURE k-mers(tour, integer k) is
2:           L ← length(tour)
3:   arr ← new array of L-k+1
4: //iterate over the number of k-mers in tour
5: //storing the nth k-mer in the output array for
6: n ← 0 to L-K+1 exclusive do
7: arr[n] ← sub tour of tour from city n inclusive to city n + k
   exclusive
8: Return arr;
9: //LOCAL SEARCH
10: GENERATE QUERY TOUR
11: REPEAT
12: GENERATE A SET OF NEW QUERY TOURS
13: EVALUATE THE NEW QUERY TOURS
14: SELECT THE BEST QUERY TOUR
15: UNTIL finding the shortest path
```

The most known operators for an order-based evolutionary algorithm are the following:

Let us consider a set of the tours travelled by salesman where n represents the total number of salesman tours $T_t = \{t_i\}_{i=1}^n = \{t_i(1), \dots, t_i(n)\}$. We consider t_1 the first tour travelled by a salesman. We consider many operations as selection, crossover, mutation for finding the best order between the cities visited by the salesman.

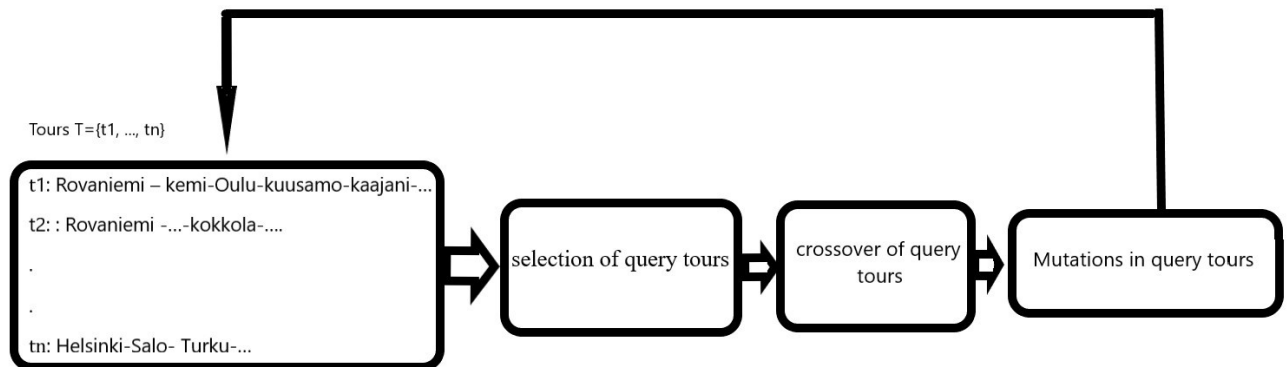


Figure 48. Order-based evolutionary algorithm for travelling salesman problem

The representation of a TSP problem is straightforwardly represented by an order-based evolutionary algorithm. It must be noted the absence of distance between the starting and departure city $C_i(n) C_i(1)$ since the alignment of first and last tours is not required.

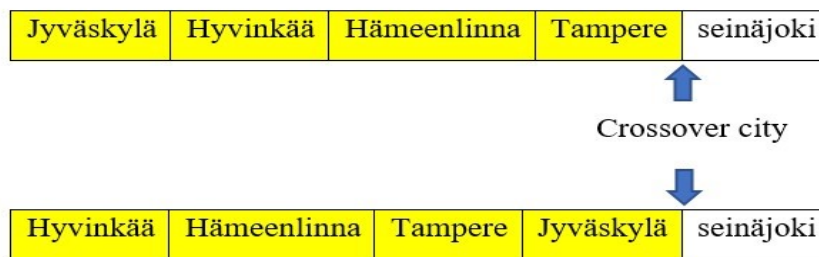


Figure 49. Crossover is performed based on single dividing point.

The selection and other evolutionary operators use to find the best tour found and where the elitist approach using zigzag scanning among the best solutions for permuting the pairs in cross over between cities [26].

```

1: Select next pair(Ci,Cj):
2:           REPEAT
3:   IF (Ci+Cj)MOD=2
4:   THEN I max(1,i-1);j j+1;
5:   ELSE THEN j max(1,j-1);i i+1;
6: UNTIL Ci≠Cj
7: RETURN (Ci+Cj);
  
```

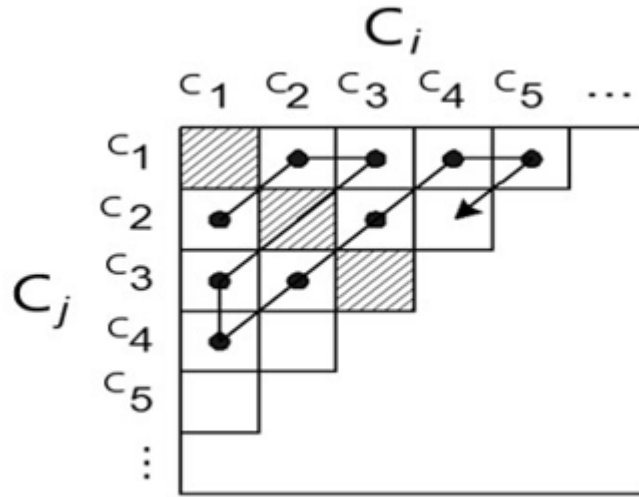


Figure 50. Zigzag scanning between the cities [26]

The transformation probabilities are called log-odds scores. Here $M_{i,j}$ is the probability that the first tour t_i transforms into the second tour t_j and P_i and P_j are the frequencies of the first and second tour. The base of the logarithm is not important, and the same substitution matrix is often expressed in different bases. For converting the probabilities to the expectation value the following relationship is used in the Equation 2 [22]:

$$t_{ij} = \log \frac{p_i M_{i,j}}{p_i p_j} = \log \frac{M_{i,j}}{p_j} = \log \frac{\text{observed frequency}}{\text{expected frequency}}$$

$$E\text{-value} \simeq m.n. P \simeq m.n. 2^{-S'} \quad \text{Equation 2}$$

In the expectation value between the tour similarity m represents the query tour length, n the database size, P the probability and S' the score bit. S' is derived from the raw alignment score defined in [22]. We consider S is a raw scoring matrix of alignment. Parameters λ and k depend on the size of tours visited by the salesman. So, the expectation value or expect value between tours similarity is more significant in lower values.

$$S' = \frac{\lambda S - \text{Log}(k)}{\text{Log}(2)}$$

Table 1. E-value interpolation [22].

E-values	Tours similarity
$E\text{-values} < 10^{-4}$	Significant homology
$10^{-4} < E\text{-values} < 10^{-2}$	Maybe not homologous
$10^{-2} < E\text{-values} < 1$	Do not indicate good homology

There is a clear relationship between tours similarity, number of local alignments and scores of Smith-Waterman Matrix. I noticed that a high tours similarity has the lowest numbers of local alignments, the minimum value of local alignments is 2 and is never less than 2 for not coinciding with global alignments as shown in Figure 51.

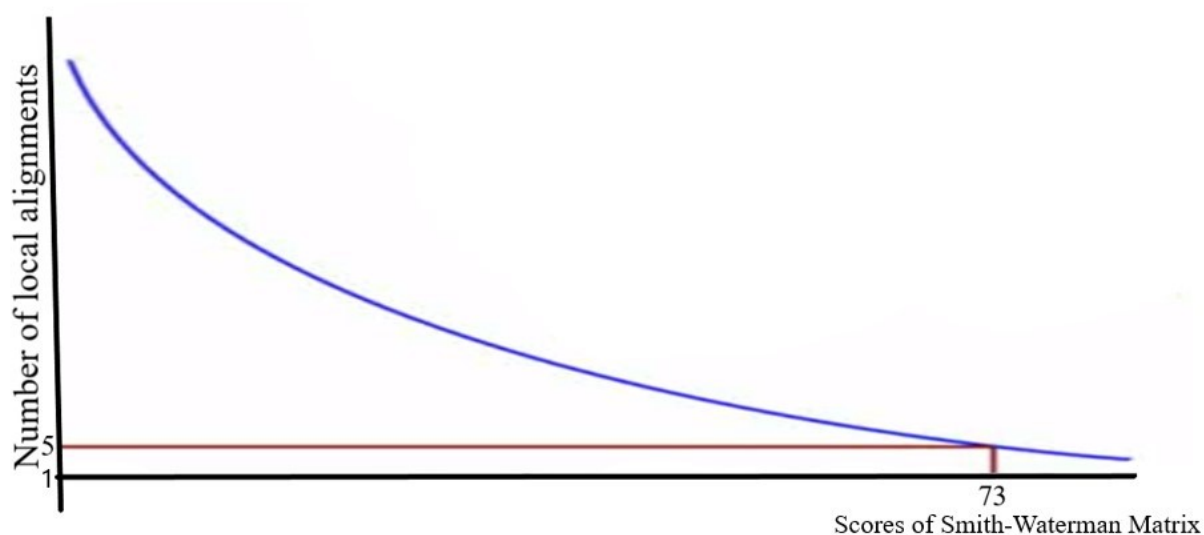


Figure 51. Local alignments property [22].



Figure 52. . Tours similarity of different local alignments [10]

5. Conclusion

The objective of the thesis was to study the different kinds of distances to classify several families for the different applications in MOPSI.

We classified the kinds of families of distances to: geographical distances and sequential distances. This classification helps to solve different problems as the distances among words during the navigation between the web pages and text editing. Benefits of this classification showed clearly when we are studying travelling salesman problem by finding the approximation solution using different kinds of distances.

Using the geographical distances, we found that the minimum spanning tree can be used to find a heuristic solution to TSP. In travelling salesman problem, Kruskal algorithm is modified to merge two spanning trees to increase the total cost of the spanning.

As well, in linear or near-linear graphs it is better to use Manhattan distance than Euclidean distance and Chebyshev distance because the L_p spaces are function spaces defined using a natural generalization of the p -norm for finitedimensional vector spaces. It will therefore reduce the time complexity of the algorithm designed.

Using the distance between sequence, or what we call in the thesis: multiple tours alignments we can find an approximative solution to the travelling salesman problem by order-based evolutionary algorithm. Based on the operations like selection, crossover, and mutations, we can update the query tours of the salesman for finding an approximate of the shortest path that provides heuristics solution to the problem.

For future work, I am interested to work about the programming of the evolutionary algorithm for finding an optimal tour for travelling of the salesman using O-MOPSI database.

References

- [1] Radu Mariescu-Istodor and Pasi Fränti. 2017. "Grid-Based Method for GPS Route Analysis for Retrieval", *ACM Trans. Spatial Algorithms Syst.* 3, 3, Article 8 (September 2017), 28 pages.
To link to this article: <http://cs.uef.fi/sipu/pub/Grid-ACM-TSAS-2017.pdf> (27.2.2020).
- [2] Radu Mariescu-Istodor and Pasi Fränti, "CellNet: Inferring Road Networks from GPS Trajectories." *ACM Trans. Spatial Algorithms Syst.* 4, 3, Article 8 (September 2018), 22 pages.
To link to this article: <http://cs.uef.fi/sipu/pub/CellNet-ACM-2018.pdf> (27.2.2020).
- [3] Radu Mariescu-Istodor, Roxana Ungureanu and Pasi Fränti, "Real-time destination prediction for mobile users." *International Cartographic Association. 15th International Conference on Location Based Services*, 11–13 November 2019
To link to this article:
https://www.researchgate.net/publication/337068715_Realtime_destination_prediction_for_mobile_users
(27.2.2020)
- [4] Radu Mariescu-Istodor and Pasi Fränti, "Gesture Input for GPS Route Search." *Springer International Publishing AG* 2016 pp. 439–449, 2016.
To link to this article: <http://cs.uef.fi/sipu/pub/GestureSearch.pdf> (27.2.2020)
- [5] Pasi Fränti, Radu Mariescu-Istodor, and Lahari Sengupta, "O-Mopsi: Mobile orienteering game for sightseeing, exercising, and education." *ACM Trans. Multimedia Comput. Commun. Appl.* 13, 4, Article 56 (August 2017), 25 pages
To link to this article: <http://cs.uef.fi/sipu/pub/O-Mopsi-TOMM.pdf> (27.2.2020)
- [6] Najlah Gali, Radu Mariescu-Istodor, Damien Hostettler, Pasi Fränti, "Framework for syntactic string similarity measure." *Expert Systems with Applications* Volume 129, 1 September 2019, Pages 169-185
To link to this article: <http://cs.uef.fi/sipu/pub/StringSimilarity-ESWA-2019.pdf> (27.2.2020)
- [7] Sami Sieranoja and Pasi Fränti. 2018. "Constructing a High-Dimensional kNN-Graph Using a Z-Order Curve." *J. Exp. Algorithmics* 23, 1, Article 1.9 (October 2018), 21 pages.
To link to this article: <http://cs.uef.fi/sipu/pub/Z-order-KNN-2018.pdf> (27.2.2020)
- [8] Haneen Arafat Abu Alfeilat, Ahmad B.A. Hassanat, Omar Lasassmeh, Ahmad S. Tarawneh, Mahmoud Bashir Alhasanat, Hamzeh S. Eyal Salman, and V.B. Surya Prasath, "Effects of Distance Measure Choice on K-Nearest Neighbor Classifier Performance: A Review" *Big Data Data science and training* Vol. 7, No. 4, 221-248, Dec 2019
To link to this article: <http://doi.org/10.1089/big.2018.0175>(27.2.2020)
- [9] Pasi Fränti, Radu Mariescu-Istodor and Karol Waga, "Similarity of mobile users based on sparse location history" *Int. Conf. Artificial Intelligence and Soft Computing (ICAISC)* Zakopane, Poland, 593-603, June 2018.
To link to this article: <http://cs.uef.fi/sipu/pub/LocationSimilarity.pdf> (27.2.2020)

- [10] Radu Mariescu-Istodor, Abu S.M. Sayem and Pasi Fränti, “Active event recommendation and attendance prediction”, *Journal of Location Based Services*, 13 (4), 201 To link to this article:<http://cs.uef.fi/sipu/pub/ActivityEventRecommendation.pdf>(27.2.2020)
- [11] Lahari Sengupta and Pasi Fränti, “Predicting the difficulty of TSP instances using MST” *IEEE International Conference on Industrial Informatics*, INDIN’19, 847-852, Helsinki 2019. To link to this article: <http://cs.uef.fi/sipu/pub/MST-knots-INDIN-2019.pdf> (27.2.2020)
- [12] Michel-Marie Deza Elena Deza. “Dictionary of Distances” *1st Edition 3rd October 2006*.
- [13] Nikhil Baba , Big Data Zone “The Levenshtein Distance Algorithm” *2 nd October 2018* .
- [14] Gregory V. Bard “Spelling-error tolerant, order-independent passphrases via the damerau-levenshtein string-edit distance metric ACSW’07”: *Proceedings of the fifth Australasian symposium on ACSW frontiers* - Volume 68 January 2007 Pages 117–124 To link to this article: <https://dl.acm.org/doi/10.5555/1274531.1274545> (27.2.2020)
- [15] Eric Robertson and Derek Yeager, Geowave, “How Space Filling curves accelerate ingest and query of geospatial data” To link to the Presentation: <https://slideplayer.com/slide/9477723/> (27.2.2020)
- [16] Andrew Hulbert , “Scaling Spatio-Temporal Analytics with GeoMesa”.
To link to the course:
https://www.geomesa.org/assets/outreach/md_datascience_final.pdf (14.4.2020)
- [17] Peer Itsik, “pairwise distance” 01-01-2001
URL:<http://www.cs.tau.ac.il/~rshamir/algmb/00/scribe00/html/lec08/node17.html#lec08:Fig:DistTree>(14.4.2020)
- [18] Radu Mariescu-Istodor, Abu S. M. Sayem & Pasi Fränti (2019): “Activity event recommendation and attendance prediction”, *Journal of Location Based Services*, Pages 293-319 Published online: 16 Sep 2019 To link to this article: <https://doi.org/10.1080/17489725.2019.1660423> (14.4.2020)
- [19] C.W. Tan, G.I. Webb, F. Petitjean, P. Reichl (September 2017): “Machine learning approaches for tamping effectiveness prediction” Faculty of Information Technology and Institute of Railway Technology Monash University, Melbourne, Australia , *Conference: 2017 International Heavy Haul Association Conference (IHHA) At: Cape Town, South Africa*

To link to this article: https://www.researchgate.net/figure/A-comparison-of-Euclidean-distance-a-with-dynamic-time-warping-b-for-time-series_fig1_317662829 (14.4.2020)
- [20] July Diana Banda Tapia, Yván Jesús Túpac Valdivia, Juan Herbert Chuctaya Humari “Optimizing Multiple Sequence Alignments using Traveling Salesman Problem and Order-based Evolutionary Algorithms” School of Computer Science, San Pablo Catholic University and Cátedra Concytec en TICs, San Agustín National University Arequipa, Peru, *Proceedings of CIBB 2012*
To link to this article: <http://rics.ucsp.edu.pe/publicaciones/20120008.pdf> (14.4.2020)

- [21] Pinky Vincent “Global Alignment algorithm with example and application” Published on July17,2012
To link to the course page: <https://www.slideshare.net/sheetalvincent/global-alignment> (14.4.2020)
- [22] Tom Madden “BLAST help NCBI help manual” *National Center for Biotechnology Information* January 28, 2011
The book online :<https://www.ncbi.nlm.nih.gov/books/NBK62051/>(14.4.2020)
- [23] Free Apps For me, 15 Best Measure Distance Apps for Android & iOS Internet
URL: <https://freeappsforme.com/measure-distance-apps/> (14.4.2020)
- [24] Pasi Fränti, Henrik Nenonen “Modifying Kruskal algorithm to solve open loop TSP”, *9th Multidisciplinary International Conference on Scheduling: Theory and Applications, (MISTA 2019)* 12-15 December 2019 Ningbo, China
To link to this article: <http://cs.uef.fi/sipu/pub/Kruskal-TSP.pdf> (14.4.2020)
- [25] Pasi Fränti, Dynamic programming ,Design and analysis of algorithm, 29.10.2018
To link to the course page: <http://cs.uef.fi/pages/franti/asa/notes.html> (14.4.2020)
- [26] Pasi Fränti, Genetic algorithms ,Advanced Topics in Algorithms , 7.4.2016
To link to the course page: <http://cs.uef.fi/pages/franti/daa++/> (14.4.2020)
- [27] Machine learning group, O-MOPSI data platform, university of eastern Finland, School of Computing Joensuu campus.
URL: <http://cs.uef.fi/mopsi/> (27.4.2020)
- [28] Victor Lavrenko and Nigel Goddard , Introductory Applied in Machine Learning course Nearest Neighbour Methods ,School of Informatics, University of Edinburgh, United Kingdom
URL: <http://www.inf.ed.ac.uk/teaching/courses/iaml/slides/knn-2x2.pdf> (11.5.2020)