UNIVERSITY OF TARTU
FACULTY OF SCIENCE AND TECHNOLOGY
INSTITUTE OF MATHEMATICS AND STATISTICS

Reigo Hendrikson

Using Gromov-Wasserstein distance to explore sets of networks

Master thesis (30 EAP)

Supervisor: PhD Raul Vicente Zafra

Tartu 2016

Using Gromov-Wasserstein distance to explore sets of networks

In many fields such as social sciences or biology, relations between data or variables are presented as networks. To compare these networks, a meaningful notion of distance between networks is highly desired. The aim of this Master thesis is to study, implement, and apply one Gromov-Wasserstein type of distance introduced by F.Mémoli (2011) in his paper ”Gromov–Wasserstein Distances and the Metric Approach to Object Matching” to study sets of complex networks. Taking into account theoretical underpinnings introduced in this paper we represent some real world networks as metric measure spaces and compare them on basis of Gromov-Wasserstein distance.

Keywords: *distance, Gromov-Hausdorff, Gromov-Wasserstein, networks, metric measure space*

Võrgustike kogumite uurimine Gromov-Wasserstein'i kauguse abil

Leidub palju valdkondi, kus andmestikud on esitatavad võrgustikena (nt. sotsiaalteadused ja bioloogia). Kahe võrgustiku vahelise sarnasuse mõõtmiseks on vajalik määrata sisukas võrgustike vaheline kaugus. Selle magistritöö eesmärk on uurida, implementeerida ja rakendada ühte Gromov-Wasserstein tüüpi kaugust, mida tutvustas F.Memoli (2011) artiklis *”Gromov–Wasserstein Distances and the Metric Approach to Object Matching”*. Antud kaugus määrab arvuliselt objektide, mis on esitatud meetriliste ruumidena, millel on defineeritud mõõt (*metric measure spaces*), vahelise kauguse/sarnasuse. Käesolevas töös esitame praktilistes andmetes esinevaid võrgustikke kirjeldatud ruumidena ning võrdleme neid Gromov-Wassersteini kauguse põhjal.

Märksõnad: *kaugus, Gromov-Hausdorff, Gromov-Wasserstein, võrgustik, mõõduga meetriline ruum*

Contents

Introduction

It is estimated that the amount of data produced doubles in size every two years. Hence, it seems fair to assume that the importance of data analysis will increase jointly with the volume of data available for researchers.

Nowadays everything is a data source. We can easily download tweets using specific software or track our heart rate without interrupting our day to day businesses. Data obtained from these sources have different structures and sophisticated methods may be needed to extract information from them. Often the final goal is to compare some objects. In order to compare objects (are two objects similar or not), one usually needs to define a notion of distance/similarity between those objects.

In many fields, including social sciences and biology, many data sets are often presented as networks. Complex networks with non-trivial topological features such as small-world or scale-free characteristics have been found in networks of human interactions, internet, or metabolic networks, to put a few examples. However, it is unclear which measure is more appropriate for comparing two or more networks. For example, it is difficult to compare two networks when the number of nodes is different or more generally, when we lack a correspondence between the nodes of one network and the nodes of the other. In the past researchers have compared global statistics of networks such as their clustering coefficient, their mean degree, or their average shortest path length. However, ideally one would like to compare different networks with a measure that takes into account their internal structures of distance/similarity between nodes. In another words, one would like to come with a measure of distance between the set of distances defined inside different networks.

The present work focuses on *Gromov-Wasserstein* (GW) distance which allows to compare objects presented as *metric measure spaces*. Current work is built on the paper by F. Mémoli [7], where different Gromov-Wasserstein type of distances between objects are defined. At the end of the paper a computational technique is introduced. This thesis focuses on implementing and applying this technique to study real world network data sets.

The contributions of this thesis are:

1) we implement the computational technique introduced in [7] in the widely used R programming language [11]. To our knowledge this is the first open implementation of the algorithm and we have uploaded it to the Github repository so that anyone can make use of our implementation. We also give an estimation of memory requirements one faces when estimating GW distance;

2) we apply GW distance together with clustering methods to real world network data sets. By that we help to understand the possibilities of GW distance to discover patterns in multiple networks' data sets.

The thesis is divided into two chapters. First chapter is devoted to methods and background theory. Second chapter describes the results of applying the analysis to real data.

First chapter starts with general definitions like *distance* and *metric* and move gradually toward definition of GW distance. After that we describe in detail computational technique used for estimating distance (similarity) between structures called metric measure spaces using GW distance. Since these techniques can be demanding in computational point of view, we also give an estimation of computational needs concerning our implementation. Lastly, we give overview of data sets used in chapter 2.

In second chapter we apply GW distance to different data sets. We use a 3D Objects data set example from [7, p. 469] to validate our implementation. Next we use GW distance in combination with clustering and dimensionality reduction methods to explore (dis)similarities between the shapes of *caenorhabditis elegans* (roundworm) neurons. We continue examining the possibilities of GW distance by applying it on Newcomb Fraternity data [10] and MIT Social Evolution data [5]. Finally we examine world trade flows [3] in 1963-2000.

Chapter 1

Methods

1.1 Distance

The authors of the *Dictionary of Distances* [2] start their book preface with the following sentence:

> *"The concept of distance is one of the basic ones in the whole of human experience. In everyday life it usually means some degree of closeness between two physical objects or ideas, i.e., length, time interval, gap, rank difference, coolness or remoteness, while the term metric is often used as a standard for a measurement."*

They also acknowledge that the number of distance metrics is infinite and argue that number of worldwide web entries offered by Google on the topic *distance* approach to 300 million. Today[1] this number is more than four times bigger.

This section intends to give a brief overview of some of the most common distances (metrics) to guide the reader to better understand Gromov-Hausdorff and Gromov-Wassestein distances discussed on section 1.2. Definitions and mathematical concepts in this chapter are taken from [2] if not stated otherwise.

We start with the mathematical definition of the terms *distance* and *metric*.

Definition 1. *Let X be a set. A function $d : X \times X \to \mathbb{R}$ is called* ***distance*** *(or* ***dissimilarity****) on X if, for all $x, y \in X$, it holds:*

[1] November 5th, 2015.

1. $d(x, y) \geq 0$ *(non-negativity);*

2. $d(x, y) = d(y, x)$ *(symmetry);*

3. $d(x, x) = 0$.

Definition 2. *Let X be a set. A function $d : X \times X \to \mathbb{R}$ is called* ***metric*** *on X if, for all $x, y, z \in X$, it holds:*

1. $d(x, y) \geq 0$ *(non-negativity);*

2. $d(x, y) = 0$ *if and only if* $x = y$*;*

3. $d(x, y) = d(y, x)$ *(symmetry);*

4. $d(x, y) \leq d(x, z) + d(z, y)$ *(**triangle inequality**).*

As we see, the latter definition has one extra requirement called triangle inequality and the second requirement is stricter than combination of first and third requirement of definition 1.

Example 1. *Not every distance is metric. Let $X = \{x_1, x_2, x_3\}$ and function $d : X \times X \to \mathbb{R}$ such that after computing $d(x_i, x_j)$ for every $i, j = 1, 2, 3$ we get matrix:*

$$d_X = \begin{pmatrix} 0 & 1 & 5 \\ 1 & 0 & 2 \\ 5 & 2 & 0 \end{pmatrix}.$$

It is easy to convince yourself that function d is distance. However we can see that function d is not metric, because triangle inequality is not satisfied:

$$5 = d(x_1, x_3) > d(x_1, x_2) + d(x_2, x_3) = 3.$$

Definition 3. *An **ultrametric** d is a metric on X which satisfies the following strengthened version of the triangle inequality:*

$$d(x, y) \leq \max\{d(x, z), d(z, y)\}$$

for all $x, y, z \in X$. So, at least two of $d(x, y)$, $d(z, y)$ and $d(x, z)$ are the same.

Next we will look at distances/metrics defined between different data structures. Specifically, we will familiarize reader with point to point, point to set and set to set distances.

1.1.1 Point to point

As the name suggests, the point-to-point distances are distances defined between two points.

Probably the most widely used point-to-point distance is Euclidean[2] distance. We can think of Euclidean distance or Euclidean metric as the straight-line distance.

Definition 4. *The **Euclidean metric** is the function $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ that assigns to any two vectors in Euclidean n-space $\mathbf{x} = (x_1, ..., x_n)$ and $\mathbf{y} = (y_1, ..., y_n)$ the number:*

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}.$$

Another example of point to point metric is city-block[3] metric.

Definition 5. *The **city-block metric** is the l_1-metric on $\mathbb{R}^2$, defined by*

$$\|x - y\| = |x_1 - y_1| + |x_2 - y_2|.$$

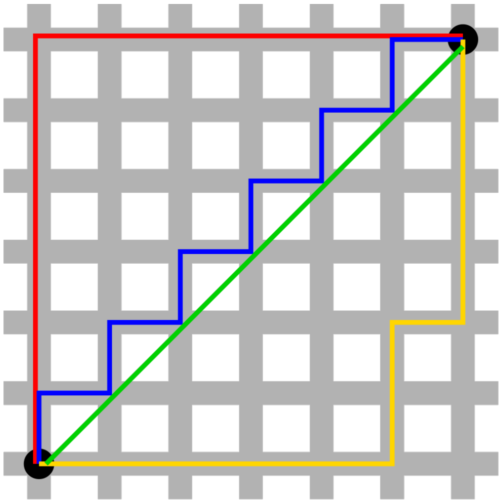


Figure 1.1: An illustration comparing the taxicab metric versus the Euclidean metric on the plane: In the taxicab metric all three pictured paths (red, yellow, and blue) have the same length (12) for the same route. In the Euclidean metric, the green path has length $6\sqrt{2} \approx 8.49$, and is the unique shortest path.[4]

[2] Also known as *Phythagorean* and *as-crow-flies* distance.

[3] Also called taxicab metric and Manhattan metric.

[4] `https://commons.wikimedia.org/wiki/File:Manhattan_distance.svg`

Figure 1.1.1 explains visually differences of the two metrics defined.

The list of metrics (distances) is endless. In this section we introduced two possibilities to define metrics between two points. We also saw that metric used has a huge effect on outcome. So one could say that it is a researcher privilege and curse to find the most appropriate function for his/her problem.

It is also clear that single points are not only objects of interest in our world. Next we will look at situations where the rational way to solve a problem is to calculate distance (or dissimilarity) between point and set of points.

1.1.2 Point to set

Section 1.1.1 gave a brief overview of point to point metrics which are easy to apprehend. When thinking of distance (dissimilarity) between objects people often reduce their problem to point to point case. In everyday use when asked what is the distance between our and our friends house we automatically think of some point to point distance (we probably use Google Maps for getting an answer). Or when some stranger stops you to ask how far is nearest supermarket from here. It is fair to assume that answers provided by Google Maps or similar systems are the "right" ones. But in mathematical sense supermarket or friend house is rather defined by set of points than a single point. So it is important to understand what exactly is meant when asked about distance or dissimilarity.

Before giving the definition of *point-set distance* we will define a structure called *metric space*.

Definition 6. *A* ***metric space*** (X, d) *is a set* X *equipped with a metric* d*.*

Example 2. *Let* $X = \{x_1, x_2\}$ *and let* $d_X = \left(\begin{smallmatrix} 0 & 2 \\ 2 & 0 \end{smallmatrix}\right)$ *be a metric on* X*. Then* (X, d_X) *is a metric space.*

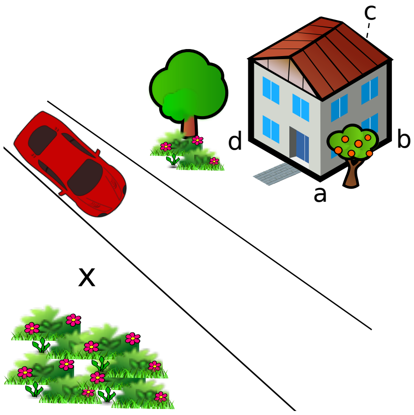
Definition 7. *Given a metric space* (X, d)*, the* ***point-set distance*** $d(x, A)$ *between a point* $x \in X$ *and a subset* A *of* X *is defined as*

$$\inf_{y \in A} d(x, y).$$

For any $x, y \in X$ *and for any non-empty subset* $A \in X$ *we have the following version of triangle inequality:* $d(x, A) \leq d(x, y) + d(y, A)$

Example 3. *Lets look at case where we want to know how far is our house from some point in the sidewalk. See picture*[5][6] *beside text. Let* $X = \{x, a, b, c, d\}$ *where* x *is a point in the sidewalk and points* $a, b, c, d \in A$ *are points defining our house. First one has to decide which metric to use. Lets use Euclidean metric such that we get*

$$d_X = \begin{array}{c|ccccc} & x & a & b & c & d \\ \hline x & 0 & 30 & 38 & 39 & 29 \\ a & 30 & 0 & 10 & 14 & 10 \\ b & 38 & 10 & 0 & 10 & 14 \\ c & 39 & 14 & 10 & 0 & 10 \\ d & 29 & 10 & 14 & 10 & 0 \end{array}.$$



Now we have the metric space (X, d_X)*. Following definition 7 we get point-set distance*

$$d(x, A) = \min\{d(x, a), d(x, b), d(x, c), d(x, d)\} = 29.$$

Now after seeing mathematical definition of point-set distance it is clear that the distance from a point in sidewalk to the nearest supermarket depends heavily on the points we use when defining the supermarket. Also we see that in discrete case infimum is replaced with minimum.

In the next section 1.1.3 we define *set-to-set distance* and introduce one of the most widely used set-to-set distances, the so-called *Hausdorff distance.*

1.1.3 Set to set

Section 1.1.2 focused on *point-set distance.* We start this section with definition of *set-set distance.*

Definition 8. *Given a metric space* (X, d)*, the* ***set-set distance*** *between two subsets* A *and* B *on* X *is defined by*

$$\inf_{x \in A, y \in B} d(x, y).$$

[5] http://res.freestockphotos.biz/vectors/16/16114-illustration-of-a-house-ve.svg

[6] https://pixabay.com/static/uploads/photo/2013/07/12/14/55/racing-car-149034_960_720.png

Point-set distance is special case of *set-set* distance where subset A (or B) has no more than one point. Also we see that the *set-set* distance would be zero if two sets intersect or have a common point.

One of most widely used set to set distance is the *Hausdorff distance* defined as:

Definition 9. *Let X and Y be two non-empty subsets of metric space (M,d).* ***Hausdorff distance*** $d_H(X,Y)$ *is defined by value*

$$d_H(X,Y) = \max\left\{\sup_{x\in X}\inf_{y\in Y} d(x,y), \sup_{y\in Y}\inf_{x\in X} d(x,y)\right\},$$

where sup represents supremum and inf infinum.

Figure 1.2 explains visually the concept of Hausdorff distance between the green line X and the blue line Y.

Informally, two sets are close in the Hausdorff distance if every point of either set is close to some point of the other set. The Hausdorff distance is the longest distance you can be forced to travel by an adversary who chooses a point in one of the two sets, from where you then must travel to the other set. In other words, it is the greatest of all the distances from a point in one set to the closest point in the other set.[17]

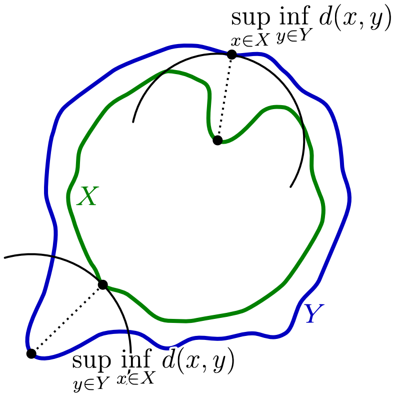

$\sup_{x\in X}\inf_{y\in Y} d(x,y)$
X
Y
$\sup_{y\in Y}\inf_{x\in X} d(x,y)$

Figure 1.2: Example of Hausdorff distance.[7]

The purpose of this section 1.1 has been to show readers a variety of distances/metrics available. Another goal has been to familiarize the reader with notions of distance between structures more complex than points.

[7] https://commons.wikimedia.org/wiki/File:Hausdorff_distance_sample.svg

Section 1.2 focuses on Gromow-Wasserstein distances which allow to measure distance/similarity between structures called *metric-measure spaces.*

1.2 Gromov-Wasserstein distance

In this section we have a closer look at Gromov-Wasserstein (GW) distance - the backbone of this thesis. We start by relevant definitions and move step by step toward the definition of GW distance. After giving all necessary definitions we look at computational aspects of GW distance. Definitions in this section are taken from [7].

We start with the definition of a *metric-measure space.*

Definition 10. *A* ***metric measure space*** *or mm-space for short is a triple (X, d_X, μ_X) where (X, d_X) is a metric space and μ_X is a Borel probability measure on X.*

In the finite case, μ_X reduces to a collection of non-negative weights, one for each point $x \in X$, such that the sum of all weights equals 1. The intuitive interpretation of $\mu_X(x)$ is that it measures the *importance* of x: points with zero weights should not matter, points with lower values of the weight should be less prominent than points with larger values of the weight [8, p. 7].

Definition 11. *Let X and Y be sets. A function $f : X \to Y$ is* ***surjective*** *if*

$$\forall y \in Y \text{ there exists } x \in X \text{ s.t. } f(x) = y.$$

Definition 12. [7, p. 426]. *Given two metric spaces (X, d_X) and (Y, d_Y), a map $\varphi : X \to Y$ is a* ***isometry*** *if φ is surjective and*

$$d_X(x, x') = d_Y(\varphi(x), \varphi(x'))$$

for all $x, x' \in X$. If X and Y are such that there exists an isometry between them, then we say that X and Y are ***isometric****.*

Definition 13. *Subset of space is called* ***compact*** *if it is closed and bounded.*

Example of compact and not compact intervals can be found in figure 1.3.

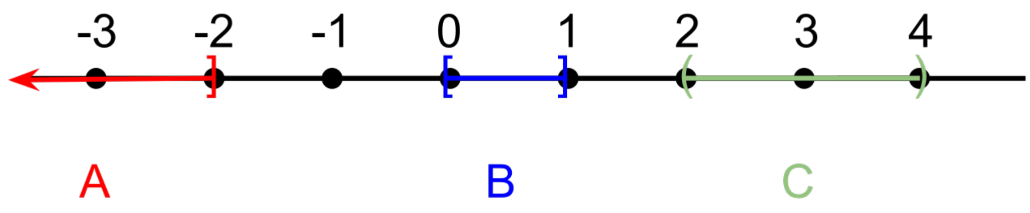


Figure 1.3: The interval $A = (-\infty, -2]$ is not compact because it is not bounded. The interval $C = (2, 4)$ is not compact because it is not closed. The interval $B = [0, 1]$ is compact because it is both closed and bounded.[8]

Definition 14. (Correspondance) *For non-empty sets A and B, a subset $R \subset A \times B$ is a* ***correspondence*** *(between A and B) if and only if*

- $\forall a \in A$ there exists $b \in B$ *s.t.* $(a, b) \in R$,
- $\forall b \in B$ there exists $a \in A$ *s.t.* $(a, b) \in R$.

Let $\mathcal{R}(A, B)$ denote the set of all possible correspondences between sets A and B.

Definition 15. (Measure coupling) *Given two metric measure spaces (X, d_X, μ_X) and (Y, d_Y, μ_Y) one says that a measure μ on the product space $X \times Y$ is a* ***coupling*** *of μ_X and μ_Y iff*

$$\mu(A \times Y) = \mu_X(A), \text{ and } \mu(X \times B) = \mu_Y(B) \tag{1.1}$$

for all measurable sets $A \subset X, B \subset Y$. Denote by $\mathcal{M}(\mu_X, \mu_Y)$ the set of all couplings of μ_X and μ_Y.

[8] https://en.wikipedia.org/wiki/Compact_space

1.2.1 From Hausdorff to Gromov-Wasserstein

This section gives all necessary definitions to comprehend the theory behind computational technique described in 1.2.2. Ideas and definitions in this section are paraphrased from [7]. Explanatory examples and comments come from author if not stated otherwise.

Hausdorff distance

We defined Hausdorff distance in section 1.1.3 (see def. 9). Hausdorff distance was defined between two non-empty subsets X and Y of metric space (M, d). In practical world there exists many cases where we want to compare objects which "live" in different spaces. One way to overcome this problem is to define a common metric space for those objects. This is exactly the idea behind *Gromov-Hausdorff* distance.

Gromow-Hausdorff distance

Let A and B denote two objects. In a nutshell, the idea of the Gromov–Hausdorff distance is that in the absence of a common metric space where both A and B are embedded, one first looks for a sufficiently rich, abstract metric space Z that admits isometric copies A' and B' of A and B, respectively. Then, a notion of dissimilarity D between A and B is computed and the arbitrariness is eliminated by optimizing over the choice of Z, where one informally calls the process by which this arbitrariness is eliminated *gromovization.* [7, p. 435]

In literature one can find many ways to define Gromov-Hausdorff distance (look [8, p. 8-12] and table 4 in [7, p. 439]). We go with the one relevant for us. For now on, for metric spaces (X, d_X) and (Y, d_Y) let

$$\Gamma_{X,Y} : X \times Y \bigtimes X \times Y \to \mathbb{R}^+$$

be given by

$$\Gamma_{X,Y}\left(x, y, x', y'\right) := \left|d_X(x, x') - d_Y(y, y')\right|.$$

Definition 16. [7, p. 437] *For compact metric spaces (X, d_X) and (Y, d_Y) the **Gromov-Hausdorff distance** is defined as*

$$d_{\mathcal{GH}}(X, Y) = \frac{1}{2} \inf_{R} \|\Gamma_{X,Y}\|_{L^\infty(R\times R)}. \tag{1.2}$$

Gromov-Wasserstein distance

The idea behind Gromov-Wasserstein distance is to substitute the L^∞ norm in (1.2) by L^p norms, and correspondences by coupling measures. The result of these substitutions will be a distance which is more amenable to practical computations but retains all desirable theoretical underpinnings. The reason why optimization becomes easier is cause we pass from a combinatorial problem to one that takes continues values. So, for $p \in [1, \infty)$ and $\mu \in \mathcal{M}(\mu_X, \mu_Y)$ (see def. 15) let

$$\begin{aligned}\mathbf{J}_p(\mu) := \frac{1}{2}\left(\int_{X\times Y}\int_{X\times Y}(\Gamma_{X,Y}(x,y,x',y'))^p \mu(dx\times dy)\mu(dx'\times dy')\right)^{1/p} \\ \left(=\frac{1}{2}\|\Gamma_{X,Y}\|_{L^p(\mu\otimes\mu)}\right)\end{aligned}$$

Definition 17. [7, p. 420] *For $1 \leq p \leq \infty$ one defines (Gromov-Wasserstein) distance $\mathfrak{D}_p$ between two mm-spaces X and Y by*

$$\mathfrak{D}_p := \inf_{\mu\in\mathcal{M}(\mu_X,\mu_Y)} \mathbf{J}_p(\mu). \tag{1.3}$$

Remark. (Relationship between (1.2) and (1.3)). Theorem 5.1 (b) in [7] asserts that $d_{\mathcal{GH}}(X, Y) \leq \mathfrak{D}_\infty(X, Y)$.

Section 1.2.2 introduces computational technique for computing distance $\mathfrak{D}_p$.

1.2.2 Computational technique

This section deals with practical computation of $\mathfrak{D}_p$. We used the computational technique described in [7, p. 466]. Algorithms leading up to estimation of $\mathfrak{D}_p$ are implemented in R programming language [11]. Functions used for solving optimization problems 1.4 and 1.6 are in appendix A and appendix B, respectively. All relevant functions are formed as R package *gwDist* which is publicly available in our `GitHub` repository[9].

Author acknowledges that this section is entirely build on the brilliance of F. Mémoli whose ideas we are presenting and who helped us to formulate alternate optimization problem 1.5.

For estimating the GW distance between two metric-measure spaces, following [7, p. 466], one has to solve optimization problem 1.4. The optimization problem for computation of $\mathfrak{D}_p$ is recast as follows:

> Assume that finite mm-spaces $\mathbb{X} = \{x_1, ..., x_n\}$ and $\mathbb{Y} = \{y_1, ..., y_n\}$ with metrics $d_{\mathbb{X}}$ and $d_{\mathbb{Y}}$, respectively, and probability measures $\mu_{\mathbb{X}}$ and $\mu_{\mathbb{Y}}$, respectively, are given. Let
>
> $$\mathcal{M} := \left\{\mu \in \mathbb{R}_+^{n_{\mathbb{X}} \times n_{\mathbb{Y}}} \middle| 0 \leq \mu_{ij} \leq 1\right\},$$
>
> where
>
> $$\begin{cases} \sum_j \mu_{i,j} = \mu_{\mathbb{X}}(x_i) \\ \sum_i \mu_{i,j} = \mu_{\mathbb{Y}}(y_i) \end{cases} \quad \text{for all } 1 \leq i \leq n_{\mathbb{X}}, 1 \leq j \leq n_{\mathbb{Y}}.$$
>
> The number of linear constraints in $\mathcal{M}$ is $(n_{\mathbb{X}} + n_{\mathbb{Y}})$. Let $p \in [1, \infty)$. Then the problem that needs to be solved is
>
> $$(P_p) \begin{cases} \min_{\mu \in \mathcal{M}} \mathbf{H}_p(\mu) \\ \mathbf{H}_p(\mu) := \sum_{i,i'=1}^{n_{\mathbb{X}}} \sum_{j,j'=1}^{n_{\mathbb{Y}}} \mu_{i,j}\mu_{i',j'} |d_{\mathbb{X}}(x_i, x_{i'}) - d_{\mathbb{Y}}(y_j, y_{j'})|^p. \end{cases} \tag{1.4}$$

Problem (P_p) is a quadratic optimization problem (QOP) with linear constraints. For solving this problem we used a technique[10] relying on solving successive linear optimization problems (LOP). After solving (P_p) we denote

[9] `https://github.com/rendrikson/gwDist`

[10] Technique described comes from e-mail exchange with F. Mémoli.

the measure coupling that one obtains upon convergence of the method with μ^* and then estimate

$$\mathfrak{D}_p(\mathbb{X}, \mathbb{Y}) \simeq \frac{1}{2}(\mathbf{H}_p(\mu^*))^{1/p}.$$

Alternate optimization problem

In order to solve 1.4 one has to solve a non-convex optimization problem with linear constraints:

$$F = \min\left\{U^T G U, \text{ over } U\right\},$$

where G is a fixed square matrix arising from $d_{\mathbb{X}}$ and $d_{\mathbb{Y}}$. U is a linearly constrained vector (in our case the constraints are the ones arising from the fact that U has to be a probability measure on the product space of two finite spaces with certain marginals). This means that

$$U = \left(\mu_{1,1}, \mu_{1,2}, ..., \mu_{n_{\mathbb{X}}, n_{\mathbb{Y}}}\right)^T.$$

For the alternate optimization problem one fixes U_0 and for each n computes

$$U_{n+1} = \arg\min\left\{U^T G U_n, \text{ over } U\right\}. \tag{1.5}$$

Eventually this process should converge to a local minimizer.

We initialize our alternate optimization problem with a solution of $(\mathbf{FLB}_p)$:

$$(\mathbf{FLB}_p) \begin{cases} \min_{\mu \in \mathcal{M}} \mathbf{L}_p(\mu) \\ \mathbf{L}_p(\mu) := \frac{1}{2} \sum_{i=1}^{n_{\mathbb{X}}} \sum_{j=1}^{n_{\mathbb{Y}}} \mu_{ij} |s_{\mathbb{X}}, p(i) - s_{\mathbb{Y},p}(j)| \end{cases} \tag{1.6}$$

where

$$s_{\mathbb{X},p}(i) := \left(\sum_{k=1}^{n_{\mathbb{X}}} \mu_{\mathbb{X}}(x_k)(d_{\mathbb{X}}(x_i, x_k))^p\right)^{\frac{1}{p}}, \text{ for } 1 \leq i \leq n_{\mathbb{X}}$$

$$s_{\mathbb{Y},p}(j) := \left(\sum_{k=1}^{n_{\mathbb{Y}}} \mu_{\mathbb{Y}}(y_k)(d_{\mathbb{Y}}(y_i, y_k))^p\right)^{\frac{1}{p}}, \text{ for } 1 \leq i \leq n_{\mathbb{Y}}.$$

Problem ($\mathbf{FLB}_p$) is LOP with linear constraints.

Figure 1.4 describes the steps involved in computing GW distance. Linear optimization problems discussed in this section were solved using R package *Rglpk*[11].

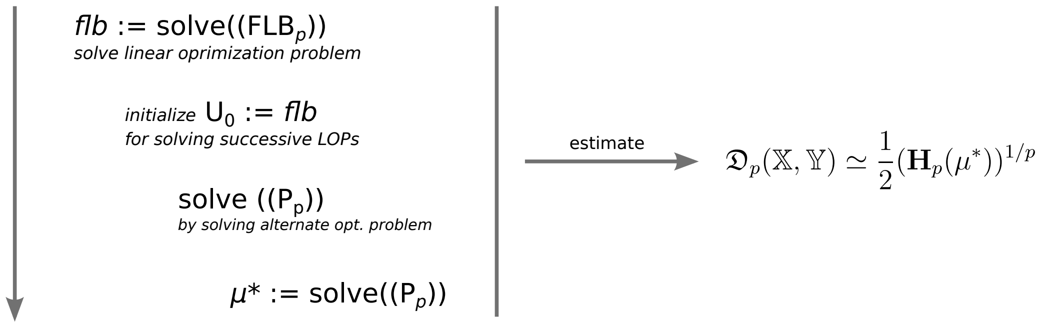
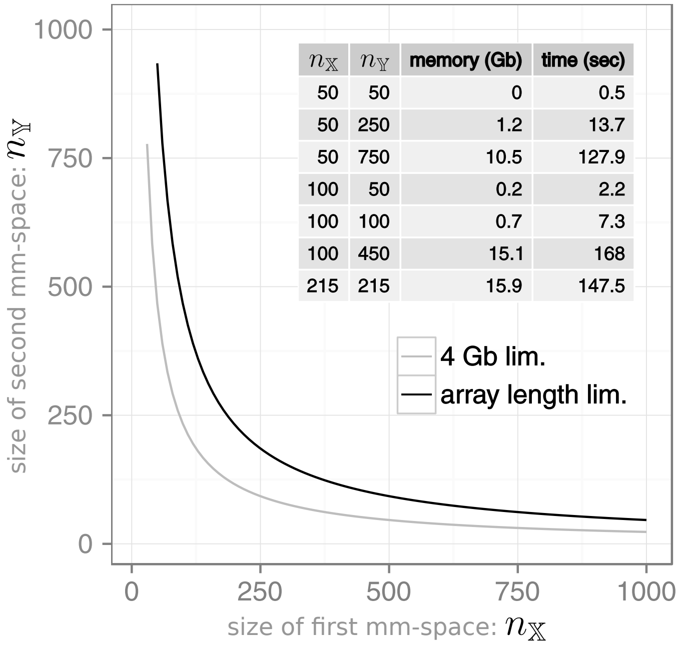

flb := solve((FLB$_p$))
solve linear oprimization problem
initialize U$_0$:= *flb*
for solving successive LOPs
solve ((P$_p$))
by solving alternate opt. problem
μ* := solve((P$_p$))
estimate
$\mathfrak{D}_p(\mathbb{X},\mathbb{Y}) \simeq \frac{1}{2}(\mathbf{H}_p(\mu^*))^{1/p}$

Figure 1.4: Algorithm flow for estimating GW distance.

[11] Rglpk is R interface to the GNU Linear Programming Kit. GLPK is open source software for solving large-scale linear programming (LP), mixed integer linear programming (MILP) and other related problems. For more information look: `https://cran.r-project.org/web/packages/Rglpk/index.html`

1.2.3 Estimation of computational needs

This section gives an overview of the computational limits our implementation of GW distance is facing. As mentioned before, we used R programming language to implement computational technique described in 1.2.2. In R the limit of each dimension of an array is set to $2^{31}-1$. For solving QOP 1.4 we solved an alternate optimization problem 1.5. Let $n_{\mathbb{X}}$ and $n_{\mathbb{Y}}$ be the number of points in mm-spaces $\mathbb{X}$ and $\mathbb{Y}$, respectively. We see that matrix G in optimization problem 1.5 has $n_{\mathbb{X}} \times n_{\mathbb{Y}}$ rows and columns. Now considering dimension limits set by R, we get that inequality $n_{\mathbb{X}}^2 \times n_{\mathbb{Y}}^2 \leq 2^{31}-1$ has to hold. We used ATLAS cluster in High Performance Computing Center of the University of Tartu for benchmarking our implementation. See figure 1.5 for details. Time in figure 1.5 is mean elapsed time (over 10 runs) for computing GW distance with function *gwDist* (see 2.2.4).



$n_{\mathbb{X}}$	$n_{\mathbb{Y}}$	memory (Gb)	time (sec)
50	50	0	0.5
50	250	1.2	13.7
50	750	10.5	127.9
100	50	0.2	2.2
100	100	0.7	7.3
100	450	15.1	168
215	215	15.9	147.5

Figure 1.5: Computational limits with respect to the sizes of metric measure spaces. Sizes of metric measure spaces are denoted with $n_{\mathbb{X}}$ and $n_{\mathbb{Y}}$. The black curve shows maximum sizes of mm-spaces such that array length limit in R is not exceeded. Gray line shows maximum sizes of mm-spaces such that memory limit of 4GB is not exceeded. Table in upright of figure gives an estimations of memory requirements and elapsed computing time of GW-distance.

1.3 Dimensionality reduction methods

For better understanding of the computed distance matrices, it it useful to visualize them. It helps us to capture emerging structures and clusters. In this section we give a brief overview of dimensionality reduction and visualization methods used in following sections.

Classical Multidimensional Scaling

Classical Multidimensional Scaling (cMDS) is a technique that displays the structure of distance-like data as a geometrical picture. In particular, it aims to find a configuration of points in a low-dimensional Euclidean space such that their inter-point Euclidean distance respects as much as possible their distance/dissimilarity in the original high-dimensional space. It is also assumed that in the distance matrix there are no missing entries, i.e., the distances between all pairs are measured. The input data for cMDS may come as similarity matrix $\boldsymbol{D}$ or as raw data point coordinates. In latter case Euclidean metric will be used to compute similarities of objects.

Sammon mapping

Sammon mapping [12] is a version of multidimensional scaling and thus another algorithm that maps a high-dimensional space to a space of lower dimensionality by trying to preserve the inter-point distances during the projection. In particular, Sammon mapping tries to give more weight (importance) to preserve the smaller distances in the original high-dimensional space. Thus, Sammon mapping is considered a non-linear approach as the mapping cannot be represented as a linear combination of the original variables. Sammon mapping aims to minimize the following error function:

$$E = \frac{1}{\sum_{i<j} d^*_{ij}} \sum_{i<j} \frac{(d^*_{ij} - d_{ij})^2}{d^*_{ij}},$$

where d^*_{ij} denotes distance between i-th and j-th objects in the original space and d_{ij} denotes the distance between their projections.

t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) [15] is a technique for dimensionality reduction. It is a nonlinear dimensionality reduction technique that is particularly well suited for embedding high-dimensional data into a space of two or three dimensions. Specifically, it models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points. Examples, references, and details about t-SNE can be found in t-SNE webpage [14].

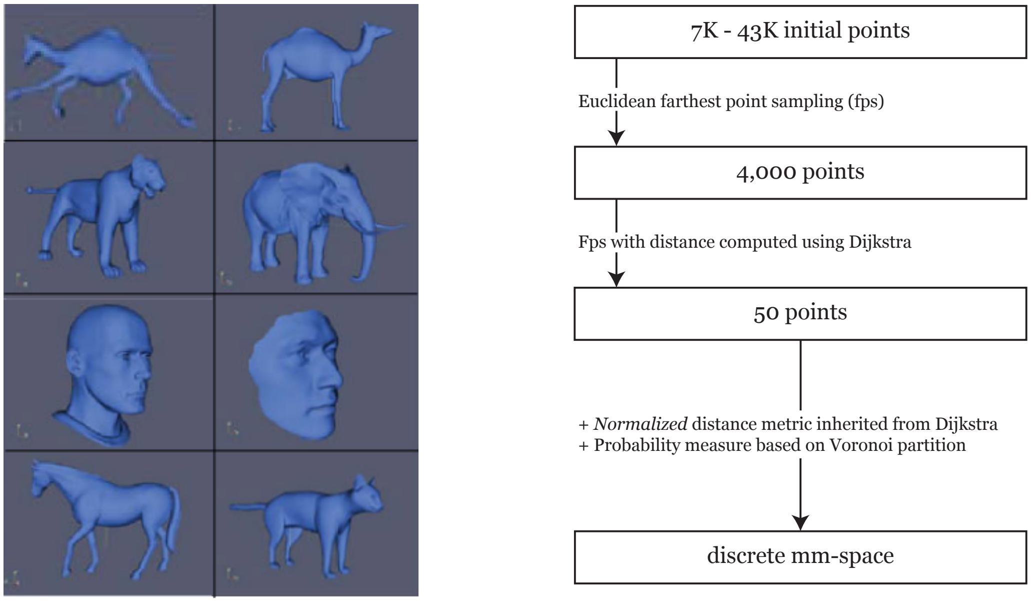
Hierarchical clustering

Hierarchical clustering is a method of cluster analysis which tries to build a hierarchy of clusters. Hierarchical clustering outputs a hierarchy (usually presented as dendrogram), a structure that is more informative than the unstructured set of cluster returned by flat clustering. A good overview of hierarchical clustering algorithms is presented in [6].

1.4 Data sets and data preparation

3D shape data

3D objects data base [13] contains 72 objects from seven different classes: *camel, cat, elephant faces heads, horse* and *lion*. Each class comprises 10-11 different *poses* on the same object. These poses are richer than just rigid isometries. For example, first row of figure 1.6a has two different poses of class *camel*. The number of vertices (points) in the models range from 7K to 43K.


7K - 43K initial points
Euclidean farthest point sampling (fps)
4,000 points
Fps with distance computed using Dijkstra
50 points
+ *Normalized* distance metric inherited from Dijkstra
+ Probability measure based on Voronoi partition
discrete mm-space

(a) poses

(b) work flow

Figure 1.6: (a)[12]: first row has two poses of class *camel*. Rows 2-4 have one pose of each class of objects. (b): diagram describing work flow from initial data to mm-spaces.

We constructed metric measure spaces by following notes in [7, p. 469]. Figure 1.6b describes the work flow for obtaining mm-spaces from initial data. We used R packages *igraph* to compute shortest paths using Dijkstra algorithm. We made some slight modifications to the function *kenStone* from R

[12] Graphics copied from [7, p. 470].

package *prospectr* when implementing a farthest point sampling. In particular, we enabled the function to use a random starting point.

Caenorhabditis elegans

Caenorhabditis elegans is free-living transparent nematode (roundworm), about 1 *mm* in length, that lives in temperate soil environments. Its simple structure, transparency, short life cycle and a small genome have made it one of the most studied organism. *C. elegans* was the first multicellular organism to have its whole genome sequenced, and as of 2012, the only organism to have its connectome (neuronal ”wiring diagram”) completed. There are 302 neurons in the nervous system of *C. elegans.*[16]

We used the computational technique described in section 1.2.2 to estimate a distance (GW distance) between the shapes of neurons of *C. elegans.* We used publicly available data from [1, 4, 9] which consists of 3D morphology reconstructions of the 302 neurons. Due to computational issues we excluded neurons named PVDR and PVDL and computed distances between the shapes of 300 neurons. Neurons in the data set have 9 to 102 sample points describing their morphology. Each sample point have a structure identifier (soma, axon, etc.), 3 spatial co-ordinates, radius, and the number of parent sample.

We used all sample points in the process of obtaining *mm-spaces* for neurons $X_k, k = 1...300$. For acquiring mm-spaces :

- we found Euclidean distance between points defined by 3 spatial coordinates;
- next we used parent sample information to define graph $G(X_k)$ with vertex set X_k;
- then we found intrinsic distance $d^{(k)}$ using Dijkstra's algorithm;
- and finally we normalized distance $d^{(k)}$ by dividing it with maximum value of $d^{(k)}$.

At this point we had metric spaces $(X_k, d^{(k)}), k = 1...300$. After that we equipped each metric space $(X_k, d^{(k)})$ with a uniform probability measure $\mu^{(k)}$. As a result we got for each neuron X_k discrete mm-space $(X_k, d^{(k)}, \mu^{(k)})$.

Newcomb fraternity

Newcomb Fraternity data [10] is publicly available data which consists of 15 matrices. Each matrix represents weekly sociometric preference rankings from 17 men attending the University of Michigan in the fall of 1956. Each participant had to rank all other 16 participants from best friend to least friend. Data was collected from week 1 to week 16 (except for week 9). First two rows of 3rd matrix can be found in table 1.1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	13	14	9	6	4	16	11	8	15	3	7	2	1	12	10	5
2	8	0	13	1	6	10	2	15	14	16	12	5	7	4	11	9	3

Table 1.1: Sample of 3rd matrix of Newcomb fraternity data. We read it as follows: in 3rd week first participant ranked 14th participant as his best friend, 13th as is second best friend etc. Second participant ranked 4th participant as his best friend, 7th as his second best friend and 10th as least friend.

Each matrix describes friendship structure of 17 participants. Our goal is to use GW distance to measure the (dis)similarity of friendship structures over the 16 weeks period. We assume that weeks that are closer to each other (e.g week 5 is closer to week 7 than week 2) have more similar friendship structures and we expect the friendship structure to crystallize after some time.

We used the *Canberra* distance for measuring (dis)similarity of subjects based on the ranks given. The idea is that people who give high and low ranks to same group of people are more similar to each other. The Canberra distance d between vectors $\mathbf{p}$ and $\mathbf{q}$ is given as follows:

$$d(\mathbf{p},\mathbf{q}) = \sum_{i=1}^{n} \frac{|p_i - q_i|}{|p_i| + |q_i|},$$

where $\mathbf{p} = (p_1, ..p_n)$ and $\mathbf{q} = (q_1, ..q_n)$. In our case $\mathbf{p}$ and $\mathbf{q}$ are e.g. two rows presented in table 1.1. Lets $M^{(k)}$ denote distance matrices computed using the Canberra distance.

Next we used *single linkage hierarchical clustering* on each matrix $M^{(k)}$. Hierarchical clustering methods give outputs in the form of dendrograms. These dendrograms can be represented as *ultrametrics*, which are a special type of metrics (see def. 3). Let $d^{(k)}, k = 1..15$ denote the obtained *ultrametrics.* We used these ultrametrics to form mm-spaces $(\mathbb{X}_k, d^{(k)}, \mu^{(k)})$, where points $\mathbb{X}_k$ are identification numbers of each participant in week k and $\mu^{(k)}$ is the uniform probability measure.

Social Evolution data

MIT Social Evolution[13] experiment [5] tracks the everyday life of a whole undergraduate dormitory with mobile phones. The Social Evolution experiment covered the locations, proximities, and phone calls of more than 80% of residents who lived in a USA university dormitory used, as captured by their cell phones from October 2008 to May 2009. This dormitory has a population of approximately 30 freshmen, 20 sophomores, 10 juniors, 10 seniors and 10 graduate student tutors.

Data collection of this experiment includes 9 data sets (check [13] for details). We used data set `proximity.csv` for our analysis.

Proximity data

Proximity data has information about bluetooth signals sent from one mobile phone to another. Data set originally consisted of 2124564 observations and 4 variables. First three rows of `proximity.csv` and description of variables are presented in table 1.2.

user.id	remote.user.id.if.known	time	prob2
58	42	2007-09-05 14:02:11	0.034
58	49	2007-09-05 14:02:11	0.000
58	54	2007-09-05 14:02:11	NA

Table 1.2: First three lines of data set `proximity.csv`. Bluetooth signal sent from whose mobile phone (*user.id*) and received by whose mobile phone (*remote.user.id*) and time, indicating the sender's mobile phone was within 10 meters of the receiver's mobile phone at the time of the record. Variable *prob2* is probability for the two person to be on the same floor in each record, estimated from Wi-Fi RSSI to access points.

We used observations between dates 29.09.2008 - 31.05.2009 and times 08:00 - 19:00. Also we excluded cases where $prob2 \leq .5$.

Our goal was to form 37 mm-spaces (one for each week). We started by finding *proximity* of subjects A and B by counting how many times a signal

[13] `http://realitycommons.media.mit.edu/socialevolution.html`. 07.10.2015

was sent from subject A phone to subject B phone or vise versa. As a result we got 37 80×80 matrices $M^{(k)}$. Thus, an element $m_{ij}^{(k)}$ represents a number of signals sent from A to B plus signals sent from B to A in week k.

Then columns and rows of matrix $M^{(k)}$ consisting of only zeros (subjects who have not been close to other subjects) were removed. By then we had 37 similarity (proximity) matrices. To get distance matrices we replaced elements $m_{ij}^{(k)}$ with $1/m_{ij}^{(k)}$. It is worth noting that relative distance changes when taking inverse. The difference between 3 & 4 and 4 & 5 is one unit. The difference between $\frac{1}{3}$ & $\frac{1}{4}$ and $\frac{1}{4}$ & $\frac{1}{5}$ is 0.08(3) and 0.05, respectively. This means that each extra signal sent or received becomes less and less important.

Next, for computational reasons infinities in the distance matrices (which emerged from dividing by 0) were replaced by $1.5 * \max$. max in this case is 1. The constant (1.5) was chosen such that distance between subjects who have not been around each other (proximity = 0) would be 150% of distance between people who have been around each other in one occasion (proximity = 1).

Later, we normalized these matrices by dividing each entry of each matrix with 1.5 (max of each matrix) and used Dijkstra algorithm for getting metrics out of these distance matrices. We denote these metrics as $d^{(k)}$.

Finally, mm-spaces $(\mathbb{X}_k, d^{(k)}, \mu^{(k)})$ were formed, where $\mathbb{X}_k$ is a set of "active" subjects identification numbers in week k and $\mu^{(k)}$ is uniform probability measure.

World trade

World trade[14] data [3] has information about world trade flows from 1962-2000. Data set has 23949 rows and 41 columns. Each row has values of importer, exporter and volume (in thousands of US dollars) for years 1962-2000 (see Table 1.3).

We reshaped this data into 39 matrices such that each matrix contains import-export information of one of the year 1962-2000. Each matrix had as many rows and columns as number of countries whose $import + export > 0$ in a given year. Our goal was to compare these matrices based on GW-distance. In order to directly use GW-distance we had to first describe the data as mm-spaces (def. 10).

[14] `http://cid.econ.ucdavis.edu/data/undata/wtf_bilat.zip`. 06.10.2015

importer	exporter	value1962	value1963	...	value2000
Estonia	Finland	-	-	...	1377611
Finland	Estonia	-	-	...	940449
Sweden	USA	316203	340684	...	4849563
USA	Sweden	169279	181301	...	9896792

Table 1.3: World trade data example.

When forming mm-spaces our starting point was that countries that have more impact to each other should be closer (in some sense) to each other. We started by computing impact of country A to country B using formula 1.7.

$$impact(A, B) = \frac{import(A, B) + import(B, A)}{allTraffic(B)}. \tag{1.7}$$

As a result we got 39 impact matrices $I^{(k)}, k = 1962...2000$ (see table 1.4).

	Estonia	Finland	Sweden	USA
Estonia	0.00	0.03	0.01	0.00
Finland	0.28	0.00	0.05	0.00
Sweden	0.12	0.09	0.00	0.01
USA	0.07	0.07	0.10	0.00

Table 1.4: Submatrix of impact matrix $I^{(2000)}$. We see that in year 2000 Finland had very big impact to Estonia. 28% of all Estonian traffic in 2000 was with Finland.

Our aim was to obtain mm-spaces from impact matrices. Specifically, impact matrices were the source of metrics.

We started by taking inverse of impact matrices, summing lower and upper matrix triangles, and dividing this sum by 2. This allowed us to get (in some sense) distances between countries.

Next we replaced infinities (emerging from taking inverse from zero) in those matrices with 1.05×(maximum value of current matrix). The constant 1.05 was chosen because the distance between countries who do not impact each other directly and the distance between countries who have very low impact to each other (in average) should be almost same.

Since the elements of these matrices had big discrepancies (some very big values) we took logarithm of all elements and normalized them by dividing with the maximum value. Let $\widehat{d}^{(k)}$ denote these distance matrices.

Next we used Dijkstra algorithm to get obtain a metric distance out of the distance matrices $\widehat{d}^{(k)}$. Finally we normalized these metrics by dividing by the maximum value of each specific metric. As a result we obtained the set of metrics $d^{(k)}$

Finally, mm-spaces $(\mathbb{X}_k, d^{(k)}, \mu^{(k)})$ were formed, where $\mathbb{X}_k$ is a set of countries (whose all import plus all export is bigger than 0) in year k and $\mu^{(k)}$ is uniform probability measure.

Chapter 2

Results

2.1 Validation on 3D Shape Objects

After implementing the computational technique described in 1.2.2 we aimed to validate our implementation by reproducing results provided in section 8.2 of [7, p. 469]. We used the publicly available triangulated objects database (see 1.4). This database consists of 72 objects. These objects represent 7 different classes: *camel, cat, elephant, faces, heads, horse* and *lion* .

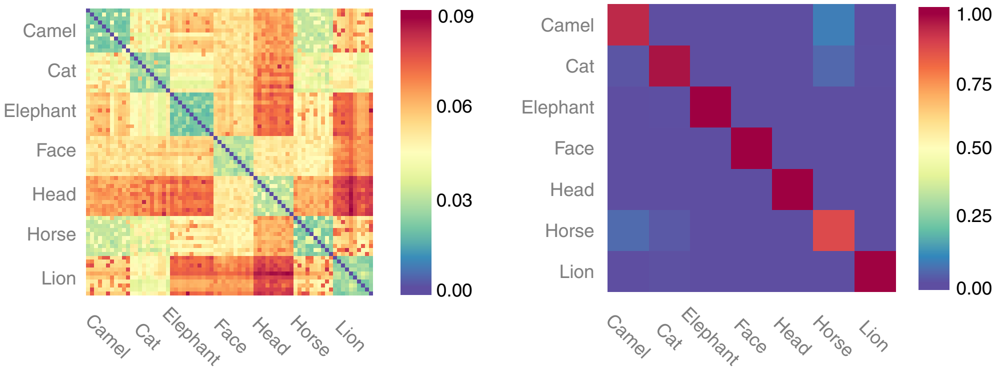


Figure 2.1: Left panel: Gromov-Wasserstein distance matrix $((d_{ij}))$. Right panel: estimated confusion matrix C for the 1-nearest neighbor classification problem.

After forming mm-spaces (see Methods section 1.4) we computed the matrix $((d_{ij}))$ such that $d_{ij} = \mathfrak{D}_1(\mathbb{X}_i, \mathbb{Y}_j)$ and solved the same classification task as in [7, p. 470]. Left panel of figure 2.1 represents matrix $((d_{ij}))$ as estimated by our implementation. Right panel of figure 2.1 shows our confusion matrix C, where C_{ij} equals the probability that the classifier will assign class j to an object when the actual class was i. Results reflecting classification power of computed distances are in table 2.1.

	$P_e(((d_{ij})))$	$P_e(\mathbf{FLB}_1)$
Original paper	0.025	0.141
Our results	0.029	0.231

Table 2.1: Comparison of probability of misclassification (P_e). "Original paper" indicates to results provided in [7]. "Our results" are results extracted from our own analysis using our implementation.

When visually comparing the matrices $((d_{ij}))$ from the original paper and from our figure 2.1 one can detect some small discrepancies. The most eye-catching is the estimated dissimilarities between elephants and camels, which in the original paper seem to be relatively big, but in our case the distances seem to be rather medium. Nevertheless, the structures of mentioned matrices look very similar.

Probability of misclassification computed on $((d_{ij}))$ (fig. 2.1) differs by 0.004 from probability of misclassification reported in original paper. Classification power of $\mathbf{FLB}_1$ in original paper outperforms our result by 0.09. See table 2.1 for more detail.

Altogether, it is fair to assume that our implementation of the computational technique described in [7, p. 466] works as it should.

2.2 Applications on real world networks

2.2.1 *Caenorhabditis elegans*

We estimated the GW distance between the shapes of 300 neurons and then applied multidimensional scaling (MDS) and hierarchical clustering algorithms to inspect for clusters in the shapes of neurons. Precise steps about how we formed metric measure spaces and the data set in general are presented in Methods section 1.4.

From the set of mm-spaces $(X_k, d^{(k)}, \mu^{(k)})$ we computed the GW distance matrix $((d_{ij}))$, such that $d_{ij} = \mathfrak{D}_1(X_i, X_j)$. After acquiring distance matrix $((d_{ij}))$ we applied hierarchical cluster analysis to explore formed structures. Figure 2.2 demonstrates a sample of neurons and the GW distance between them. We observe that neurons that look alike in visual inspection e.g. DD3

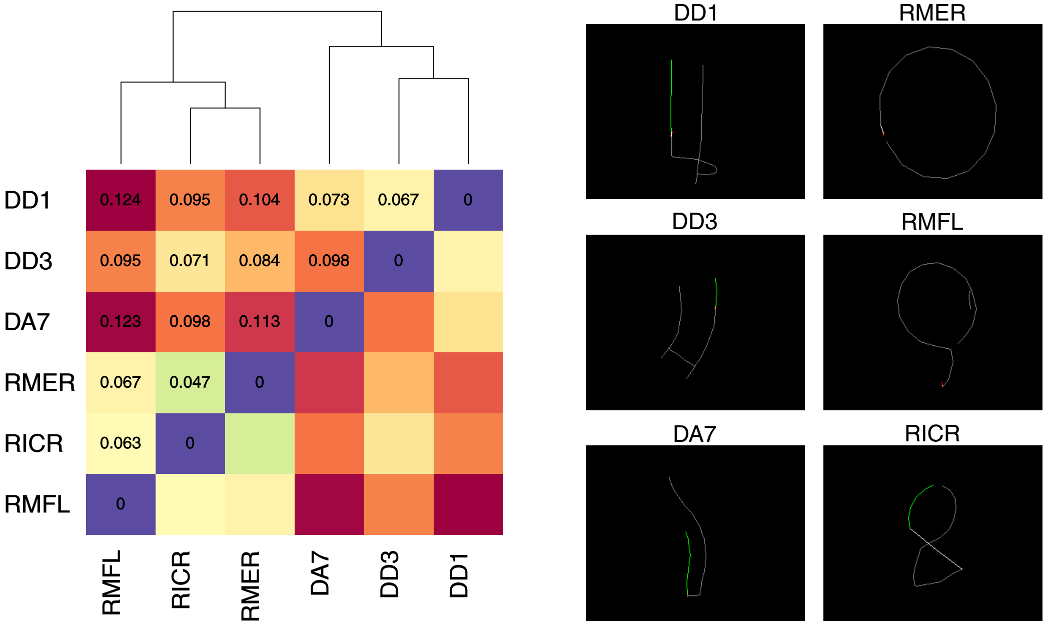


Figure 2.2: *Left panel*: GW distance between shapes of different neurons. Numbers in each shell represent GW distance between two neurons. On top of distance matrix is dendrogram showing which neurons are more similar to one another based of GW distance computed between shapes of neurons. *Right panel*: shapes of 6 neurons.

& DD1 have a smaller GW distance than neurons that do not look similar e.g RMFL & DA7. GW distance between mentioned pairs of neurons is 0.067 and 0.123, respectively.

We also compared these formed structures with known neuron groups. See figure 2.3 for details.

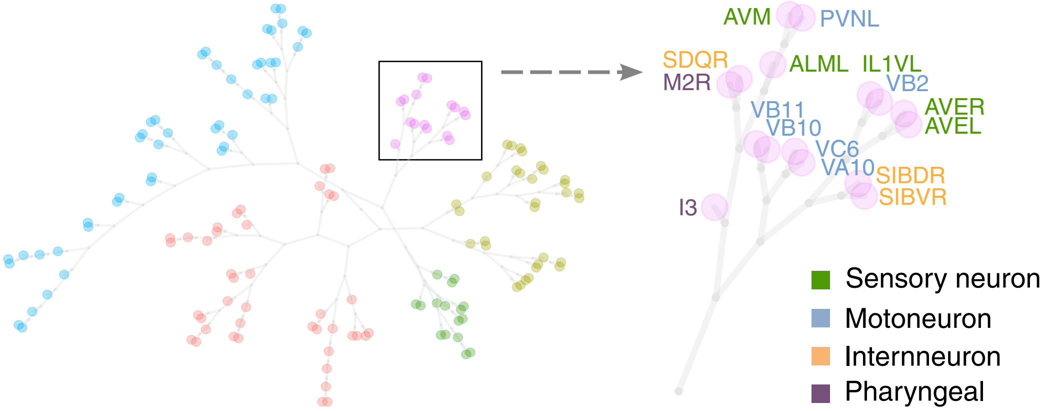


Figure 2.3: *Left*: tree of subset of 150 neurons. Five cluster using the clustering method "ward.D". *Right*: zoom-in to one of the leafs. Colors show neuron types. There are 10 types of neurons in connectome of C elegans.

Right side of figure 2.3 depicts one leaf of tree obtained using hierarchical cluster analysis. We also tried different multidimensional scaling algorithms (cMDS, MDS Sammons, t-SNE) on matrix $((d_{ij}))$ to explore emerging clusters but we did not discover any explicit cluster by visual inspection. Nevertheless, being able to compare neurons on basis of neurons shape can be a useful tool in the hands of experts.

2.2.2 Newcomb Fraternity

Newcomb Fraternity data [10] consists of 15 matrices. Each matrix represents weekly sociometric preference rankings from 17 men attending the University of Michigan in the fall of 1956. See section 1.4 for a detailed description of data and how we formed the associated metric measure spaces.

After forming a set of mm-spaces $(\mathbb{X}_k, d^{(k)}, \mu^{(k)})$ we computed the GW distance matrix $((d_{ij}))$ such that $d_{ij} = \mathfrak{D}_1(\mathbb{X}_i, \mathbb{Y}_j)$, where $1 \leq i < j \leq 17$. Left panel of figure 2.4 is a representation of $((d_{ij}))$. GW distance matrix $((d_{ij}))$ was then given as an argument to solve the *single linkage hierarchical* *clustering* problem. See right panel of figure 2.4 for a graphical representation of the dendrogram obtained.

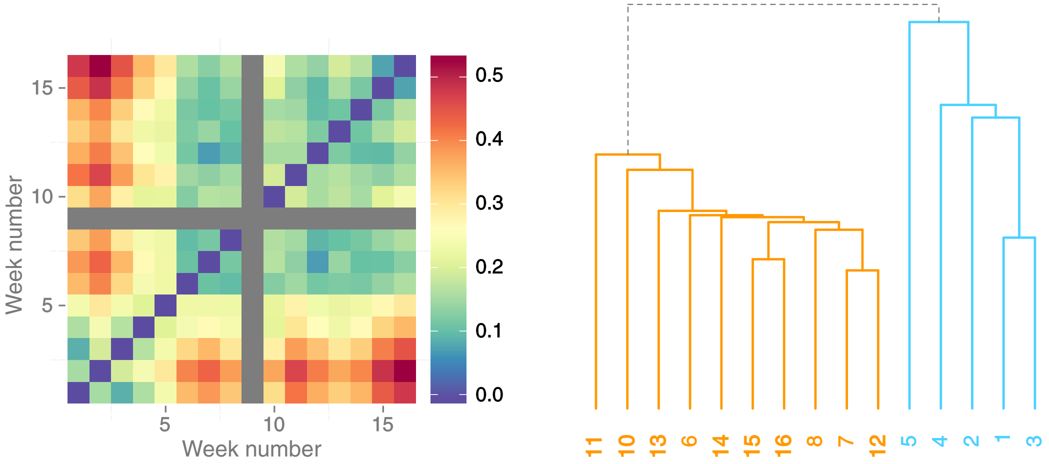


Figure 2.4: Similarity of friendship structures during 16 week period. Distance matrices based on different techniques. *Left panel*: matrix $((d_{ij}))$. *Right panel*: dendrogram of distance matrix $((d_{ij}))$.

Our hypothesis was that weeks that are closer to each other have a more similar friendship structures and we expected the friendship structure to crystallize after some time. Visual evaluation of figure 2.4 seems to confirm our hypothesis to some extent. We see that weeks 1-5 differ substantially from later weeks. Also it seems that after week 9 there was a little ”restart” possibly implying that friendship structures in subjects level were re-evaluated.

2.2.3 MIT mobility data

MIT Social Evolution experiment [5] tracks the everyday life of a whole undergraduate dormitory with mobile phones. The Social Evolution experiment covered the locations, proximities, and phone calls of dormitory residents. We used proximity data between dates 29.09.2008 - 31.05.2009 and times 08:00 - 19:00 for our analysis. Proximity data has information about bluetooth signals sent from one mobile phone to another. Data description of MIT mobility data and details about how we form the associated metric measure spaces can be found in Methods section 1.4.

After obtaining the set of metric measure spaces $(\mathbb{X}, d^{(k)}, \mu^{(k)})$ matrix $((d_{ij}))$ was computed such that $d_{ij} = \mathfrak{D}_1(\mathbb{X}_i, \mathbb{Y}_j)$. See figure 2.5 for graphical representation of $((d_{ij}))$.

To visualize any emerging clusters under our notion of distance, we used different multidimensional scaling techniques on the GW distance matrix $((d_{ij}))$. Here, we show the plot produced by using the t-SNE method. Right panel of figure 2.5 is a projection of distance matrix $((d_{ij}))$ using t-SNE algorithm.

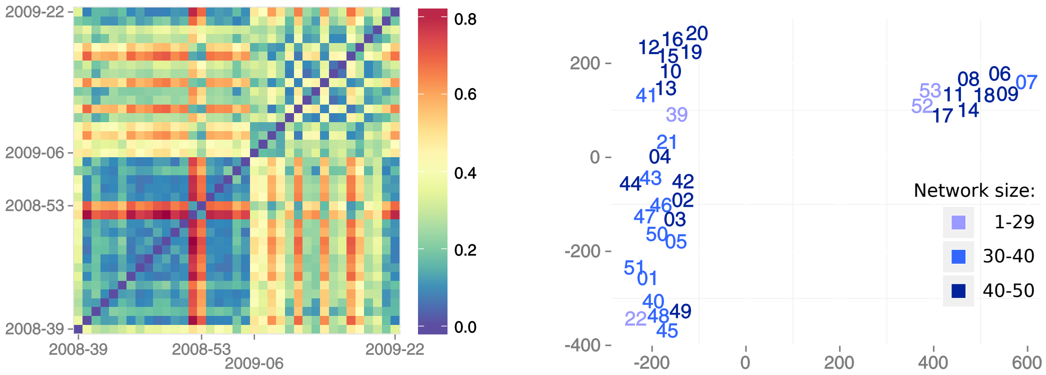


Figure 2.5: Left panel: matrix $((d_{ij}))$. Right panel: t-SNE projection of matrix $((d_{ij}))$. Numbers 39-53 represent weeks 39-53 of year 2008 and numbers 01-22 represent weeks 1-22 of year 2009.

It is also worth considering impact of the mm-space size as a contributor of forming clusters. Metric measure spaces sizes mainly range from 30-49. Outliers are weeks 39, 52, 53 and 22 by having 2,3,4 and 18 points, respectively. Network sizes are presented in right panel of figure 2.5.

Visual inspection of right panel of figure 2.5 indicates the presence of two clearly distinctive cluster. First cluster consists of weeks 52, 53, 6, 7, 8, 9, 11, 14, 17 and 18. The components of this cluster can be labeled as end of year (52, 53), beginning of new semester (6, 7, 8, 9), spring break (14) and first finals (17, 18; also Patriots day - official state holiday, was in week 17). It seems that the weeks in this cluster differ from general study weeks. Second cluster is fairly broad and we label it as ”general study weeks” cluster.

2.2.4 World trade data

World trade data contains information about world trade flows from 1962-2000. Overview of data and details concerning metric measure spaces are presented in section 1.4. The metric measure spaces $(\mathbb{X}_k, d^{(k)}, \mu^{(k)})$ were built using the GW distance matrix $((d_{ij}))$ such that $d_{ij} = \mathfrak{D}_1(\mathbb{X}_i, \mathbb{Y}_j)$. See left panel of figure 2.6 for a graphical representation of $((d_{ij}))$.

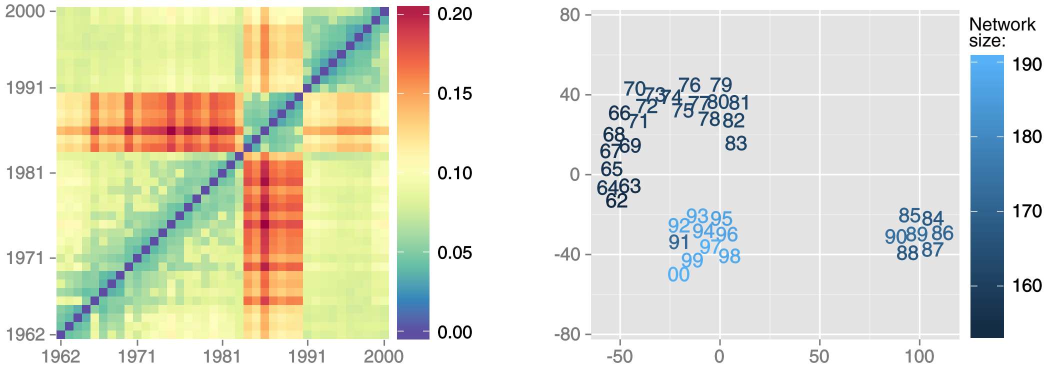


Figure 2.6: World trade flows from 1962 to 2000. *Left panel*: matrix $((d_{ij}))$. *Right panel*: projection of distance matrix $((d_{ij}))$ using t-SNE. Numbers 62-99 represent years 1962-1999. Number 00 represents year 2000.

Next, we applied cMDS, Sammon mapping, and t-SNE algorithms on distance matrix $((d_{ij}))$ to explore emerging clusters in two dimensional space. t-SNE seemed to do the best job in producing more clearly distinguishable clusters. See right panel of figure 2.6 for a graphical representation of t-SNE embeding of the distance matrix $((d_{ij}))$.

We observe that financial networks from 1962 to 2000 form three clearly distinguishable clusters:

1) years 1962 - 1983;

2) years 1984 - 1990;

3) years 1991 - 2000.

These clusters make sense since we expect consecutive years to be close to each other. One can associate these clusters with changes happening in the world. In 80s and 90s many countries obtained independence. One way to monitor the changes happening in world trade networks is to look at networks sizes. We noted that network sizes differ by clusters. It is fair to assume that additional countries also change, in many aspects, the internal structure of networks.

Nevertheless, we also note that the emergence of these clusters could be affected by data collection or by the technique used for forming the metric measure spaces. A more thorough analysis will be required to fairly interpret the results.

Discussion

In this thesis we introduced one version of the Gromov-Wasserstein (GW) distance. In particular, we implemented the Gromov-Wasserstein distance (which bounds the Gromov-Hausdorff distance) in R programming language. Given the technicalities involved in the implementation, we hope that by uploading our implementation we help to add the GW distance to the common toolkit of distances for data analysis. We also estimated the memory requirements of our implementation and benchmarked the time our implementation takes for computing GW distance between two objects.

In the second part of the thesis we applied the resulting algorithm together with visualization and clustering techniques to compare and study the structure of a set of networks.

We successfully discovered some significant patterns when applying the algorithm to biological, social, and economical data sets.

From the dimensionality reduction methods we compared for visualizing distance matrices, the t-Distributed Stochastic Neighbor Embedding (t-SNE) seems to give most desirable results (in sense of clarity of formed clusters). Nevertheless, it is important to remember that different notions of distance/dissimilarity are possible for any network and that they might change the results.

A strong limitation when using the strict formalism of GW distance is that it applies to metric measure spaces. For many networks their internal notions of dissimilarity or distance do not conform to a metric. However, numerical experiments suggest that when applied to distance matrices that do not respect the triangle inequality, the GW formalism still provides a useful notion to compare and discriminate between them.

In future, it is important to improve our implementation of GW distance so that larger networks could be compared and studied. For example, metabolic, proteomic, and other biological networks typically range in the few thousand of nodes. Once we can overcome the memory limitations set by our implementation, we will be able to compare the global internal structures of any of these types of networks for many different organisms.

Conclusion

In this thesis we have shown that Gromov-Wasserstein distance is a powerful tool for determining (dis)similarity of complex networks. When coupling this notion of distance with clustering and dimensionality reduction techniques it is possible to visualize the structure of a set of real networks. We applied our analysis to sets of networks from domains ranging from social sciences, economics, and biology.

We also noted that in many cases forming objects as metric measure spaces is not trivial or natural and may require some extra effort from researchers. We also went over the computational issues concerning our implementation which implied that computing GW distance between larger networks (>200 nodes) is computationally demanding.

In the near future it will be very interesting to apply the GW distance to larger networks, a challenge that will drive us to improve the numerical aspects of our implementation.

Bibliography

[1] G. A. Ascoli. Mobilizing the base of neuroscience data: the case of neuronal morphologies. *Nature Reviews Neuroscience*, 7(4):318–324, 2006.

[2] M.-M. Deza and E. Deza. *Dictionary of distances.* Elsevier, 2006.

[3] R. Feenstra and R. Lipsey. Nber-united nations trade data 1962-2000. `http://cid.econ.ucdavis.edu/wix.html`, 2005. [Online; accessed 17-January-2016].

[4] T. W. Harris, I. Antoshechkin, T. Bieri, D. Blasiar, J. Chan, W. J. Chen, N. De La Cruz, P. Davis, M. Duesbury, R. Fang, et al. Wormbase: a comprehensive resource for nematode research. *Nucleic acids research*, 38(suppl 1):D463–D467, 2010.

[5] A. Madan, M. Cebrian, S. Moturu, K. Farrahi, et al. Sensing the" health state" of a community. *IEEE Pervasive Computing*, (4):36–45, 2012.

[6] C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[7] F. Mémoli. Gromov–wasserstein distances and the metric approach to object matching. *Foundations of computational mathematics*, 11(4):417–487, 2011.

[8] F. Mémoli. Metric structures on datasets: stability and classification of algorithms. In *Computer Analysis of Images and Patterns*, pages 1–33. Springer, 2011.

[9] NeuroMorpho.org. `http://neuromorpho.org/`. [Online; accessed 9-November-2015].

[10] T. M. Newcomb. *The acquaintance process.* Holt, Rinehart & Winston, 1961.

[11] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2015.

[12] J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, (5):401–409, 1969.

[13] R. W. Sumner and J. Popovic. Mesh data from deformation transfer for triangle meshes. https://people.csail.mit.edu/sumner/research/deftransfer/data.html.

[14] L. van der Maaten. t-sne. https://lvdmaaten.github.io/tsne/, 2015. [Online; accessed 17-January-2016].

[15] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[16] Wikipedia. Caenorhabditis elegans — wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Caenorhabditis_elegans&oldid=687959891, 2015. [Online: accessed 5-November-2015].

[17] Wikipedia. Hausdorff distance — wikipedia, the free encyclopedia, 2015. [Online; accessed 22-December-2015].

A Function for computing GW distance

```
library(Rglpk)
gwDist <- function(initial_values,d_X,d_Y,mu_X,mu_Y, tol = 0.001, p = 1)
{
      # change in objective function
      change <- 1
      G <- mat_G(d_X = d_X, d_Y = d_Y)
      U0 <- initial_values
      res <- c()
      # first
      obj <- G %*% U0
      mat <- mu_constraints(mu_X,mu_Y)
      dir <- rep("==", nrow(mat))
      rhs <- c(mu_X,mu_Y)

      bounds <- list(lower = list(ind = c(1:ncol(mat)),
                                  val = c(rep(0,ncol(mat)))),
                     upper = list(ind = c(1:ncol(mat)),
                                  val = c(rep(1,ncol(mat)))))

      result <- Rglpk_solve_LP(obj = obj, mat = mat, dir = dir,
                               rhs = rhs, max = FALSE, bounds = bounds)
      res <- c(res,result$optimum)

      # iterative part
      while(tol < change)
      {
            obj <- G %*% result$solution
            result <- Rglpk_solve_LP(obj = obj, mat = mat, dir = dir,
                                     rhs = rhs, max = FALSE, bounds = bounds)
            res <- c(res,result$optimum)
            change <- abs(tail(res,2)[1] - tail(res,2)[2])
      }

      distance <- 0.5*H_mu(mu = result$solution, X ,Y,d_X,d_Y,mu_X,mu_Y,p = p)
      return(list("optimum" = distance, "steps" = res))
}
```

B Function for optimizing FLB

```
require(Rglpk)
?Rglpk_solve_LP
solve_FLB_Rglpk <- function(X, Y, d_X, d_Y, mu_X, mu_Y, p = 1)
{
      obj0 <- L_p(X,Y,d_X,d_Y,mu_X,mu_Y,p = p)$obj_coef
      mat0 <- mu_constraints(mu_X,mu_Y)
      dir0 <- rep("==", nrow(mat0))
      rhs0 <- c(mu_X,mu_Y)

      bounds0 <- list(lower = list(ind = c(1:ncol(mat0)),
                                  val = c(rep(0,ncol(mat0)))),
                     upper = list(ind = c(1:ncol(mat0)),
                                  val = c(rep(1,ncol(mat0)))))

      result <- Rglpk_solve_LP(obj = obj0, mat = mat0, dir = dir0,
                               rhs = rhs0, max = FALSE, bounds = bounds0)
      # status 0 ==> optimal solution found
      return(result)
}
```

C Other necessary functions

```
library(Rcpp)
Rcpp::cppFunction('NumericMatrix mat_G(NumericMatrix d_X, NumericMatrix d_Y) {
                  NumericMatrix G(d_X.nrow()*d_Y.nrow(),d_X.nrow()*d_Y.nrow());
                  for (int i = 0; i < d_X.nrow(); i++) {
                  for (int j = 0; j < d_Y.nrow(); j++) {
                  for (int ii = 0; ii < d_X.nrow(); ii++) {
                  for (int jj = 0; jj < d_Y.nrow(); jj++) {
                  G(i*d_Y.nrow()+j,ii*d_Y.nrow()+jj) = fabs(d_X(i, ii) - d_Y(j, jj));
                  };
                  };
                  };
                  };
                  return(G);
                  }
                  ')
```

```
library(Rcpp)
Rcpp::cppFunction('double H_mu_typed(NumericMatrix mu, NumericMatrix d_X,
                  NumericMatrix d_Y) {
                  double value=0;
                  value = 0;
                  for (int i = 0; i <d_X.nrow(); i++) {
                  for (int ii = 0; ii < d_X.nrow(); ii++) {
                  for (int j = 0; j < d_Y.nrow(); j++) {
                  for (int jj = 0; jj < d_Y.nrow(); jj++) {
                  value = value + mu(i, j) * mu(ii, jj) * fabs(d_X(i, ii) - d_Y(j, jj));
                  };
                  };
                  };
                  };
                  return(value);
                  }
                  ')

H_mu <- function(mu,X ,Y,d_X,d_Y,mu_X,mu_Y,p = 1)
{
      d_X <- as.matrix(d_X)
      d_Y <- as.matrix(d_Y)
      mu <- matrix(mu, nrow = nrow(d_X), byrow = TRUE)
      return(H_mu_typed(mu = mu, d_X = d_X, d_Y = d_Y))
}
```

```
mu_constraints <- function(mu_X,mu_Y)
{
      mu <- matrix(NA, nrow = length(mu_X),
                   ncol = length(mu_Y), byrow = T)
      mu_pos <- matrix(c(1:length(mu)), byrow = T,
                       nrow = length(mu_X), ncol = length(mu_Y))

      c_mat <- matrix(0,nrow = length(mu_X) + length(mu_Y),
                      ncol = length(mu))
      for(i in 1:(length(mu_X) + length(mu_Y)))
      {
            if(i <= length(mu_X))
            {
                  c_mat[i,c(mu_pos[i,])] <- 1
            }
            else
            {
                  c_mat[i,c(mu_pos[,i-length(mu_X)])] <- 1
            }
      }
      # returns matrix with n_X + n_Y rows (nr of linear constraints) and
      # n_X * n_Y columns (nr of mu_ij's)
      # each row satifies left side of one linear constraint
      # right side = mu_X(1), mu_X(2), ..., mu_Y(1), mu_Y(2), ...
      # each column marks one of mu_ij (mu_11,mu_12, mu_13, ..., mu_21, ... )
      return(c_mat)
}
```

```
L_p <- function(X,Y,d_X,d_Y,mu_X,mu_Y, p = 1)
{

      sXY <- s_XY(X,Y,d_X,d_Y,mu_X,mu_Y, p = p)
      s_X <- sXY$s_X
      s_Y <- sXY$s_Y

      S <- matrix(NA,nrow = nrow(X), ncol = nrow(Y))
      for(i in 1:nrow(X))
      {
            for(j in 1:nrow(Y))
            {
                  S[i,j] <- abs(s_X[i] - s_Y[j])
            }
      }
      S <- t(S)
      return(list("obj_coef" = c(0.5 * S)))
}
```

```
s_XY <- function(X,Y,d_X,d_Y,mu_X,mu_Y, p = 1)
{
      d_X <- as.matrix(d_X)
      s_X <- rep(NA,nrow(X))
      for(i in 1:nrow(X))
      {
            s_X[i] <- (mu_X %*% (d_X[,i])^p)^(1/p)
      }

      d_Y <- as.matrix(d_Y)
      s_Y <- rep(NA,nrow(Y))
      for(i in 1:nrow(Y))
      {
            s_Y[i] <- (mu_Y %*% (d_Y[,i])^p)^(1/p)
      }
      return(list("s_X" = s_X, "s_Y" = s_Y))
}
```

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, Reigo Hendrikson (sünnikuupäev 11.12.1990),

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Using Gromov-Wasserstein distance to explore sets of networks", mille juhendaja on Raul Vicente Zafra,

   (a) reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace'i lisamine eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

   (b) üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace'i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadustest tulenevaid õigusi.

Tartus, 18.01.2016.