technische universiteit eindhoven
Department of Mathematics and Computer Science

Master's Thesis

Geographic Grid Embeddings

by

Frank Staals

Supervisor

dr. Bettina Speckmann

Eindhoven, July 20, 2011

Abstract

We study a novel approach for visualising data with a geographic component which is inspired by, but more abstract than, spatially ordered tree maps. Given a map with regions, and a number of data attributes for each region, we wish to display the data such that we can easily compare the data values for multiple regions. At the same time we would like to get an overview of the geographical distribution of the data. Our *geographic grid embedding* represents the map by a regular grid in which spatial relations between the regions are preserved. We model the embedding of the regions into grid cells as a point set matching problem, and we give three criteria for this matching: minimising distance, preserving adjacencies, and preserving directional relations. We present methods to compute a matching that minimises the total distance under translation when using the Manhattan- or squared Euclidean distance. For the Manhattan distance we can compute a minimal distance matching under scaling as well. Furthermore, we show that the problem of preserving adjacencies is NP-hard and give a 4-approximation algorithm. Finally, we present an experimental validation of our geographic grid embeddings.

Contents

1 Introduction

At every election you see them: long lists with the results per state, county, or municipality. If you are lucky there is also a map in which colours indicate the largest party in each region. However, there is often no way to easily find and compare all results for multiple regions and get a geographical overview at the same time.

We see data with an important geographic component in almost all areas of our society. Whether it are election results, the number of cars sold in each county, or demographic profiles per state, we encounter *geo-referenced* data everywhere. As a result there is a need for proper tools and techniques to visualise geo-referenced data.

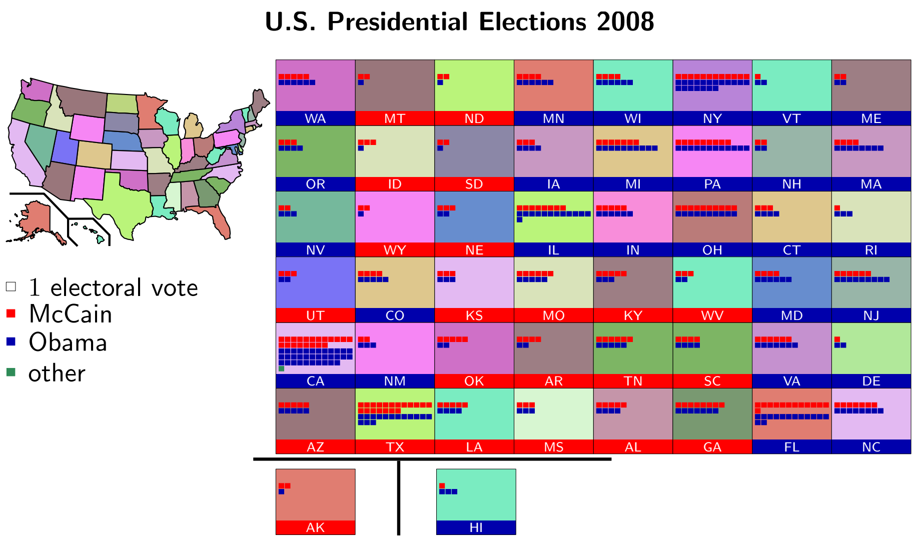


Figure 1.1: Our geographic grid embedding allows for a variety of visualisation techniques to display data for the regions – for example a bar chart-like visualisation to display election results.

Our goal is to design a new visualisation technique that can be used to display detailed information per region whilst keeping an global overview of the data. In particular we are aiming for a visualisation for geo-referenced data with the following properties:

- the visualisation should be easy to read,
- it should be easy to find the data for a given region,
- the data for different regions should be easy to compare, and
- the visualisation should be usable for both scalar values and multivariate data.

To this end we propose a visualisation technique based on the embedding of a map into a regular grid. Our method matches each region in the map to a grid cell based on the region's geographic location. The matching is designed to preserve key spatial properties of the regions, including adjacencies and directional relations. We then display the data for a region in its corresponding grid cell using a visualisation technique of our choice. We call such an embedding of a map into a grid a *geographic grid embedding.*

By embedding the map into a regular grid we get a simple and clean visualisation. Since the embedding retains important spatial properties it is relatively easy to locate the grid cell corresponding to a given region. Furthermore, our approach does not impose restrictions on how the grid cells themselves are used. This allows us to display the data in various ways and ensures that the data for different regions are easy to compare. Standard options to display the data include pie- or bar charts, but we can also use more complex visualisations such as shown in Figure 1.1. This figure shows the results of the U.S. presidential elections in 2008[1]. Each grid cell contains two areas. The upper area contains a number of small rectangles, one for each electoral vote, coloured according to the number of votes each candidate received. The lower area displays the name of the state and indicates the overall winner.

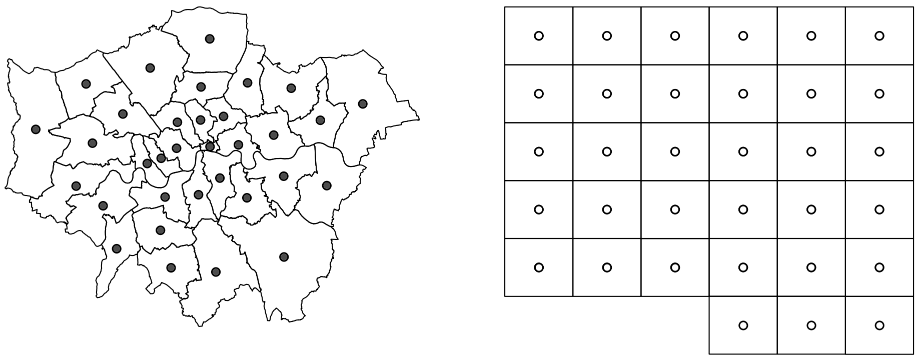


Figure 1.2: The set of blue points A representing the regions and the set of red points B representing the grid cells.

Matching regions to grid cells. For our visualisation to be effective we need a matching between regions and grid cells that preserves spatial

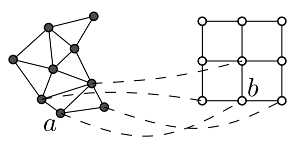
[1]Data taken from `http://www-personal.umich.edu/~mejn/election/2008/`

properties. We model this as a *one-to-one point set matching* in the plane. Each region is represented by a blue point (dark grey in greyscale) and each grid cell is represented by a red point (white in greyscale). For both the regions and the grid cells we use their centroid as representative point. Given a map and a grid we obtain a set of blue points A and a set of red points B as shown in Figure 1.2.

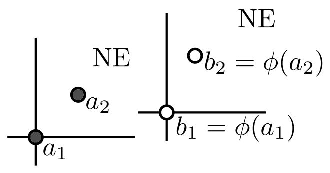
The goal now is to find the "best" one to one matching of the set of blue points A to the set of red points B. We measure the quality of a matching $\phi : A \to B$ based on the following three criteria:

$\phi(a_1)$
a_1
$\phi(a_2)$
a_2

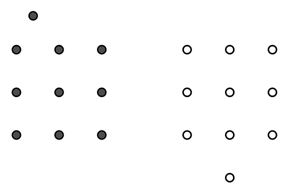
Distance. The matching should minimise the total distance between matched points.



Adjacencies. The matching should preserve the adjacencies of the regions that are represented by the points. Given the map we consider the dual graph $G = (A, E)$ in which $(a_1, a_2) \in E$ if and only if the region represented by a_1 is adjacent to the region represented by a_2. Similarly we can consider the dual graph of the grid $H = (B, Z)$. If a point $a \in A$ is mapped to point $b \in B$ then the neighbours of a should be mapped to neighbours of b.

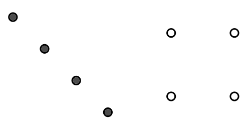


Directional Relation. The matching should preserve the directional relation between pairs of points. For example, if a point a_1 is mapped to b_1, a_2 to b_2, and a_2 is to the north-east of a_1, then b_2 should also be to the north-east of b_1.



The distance criterion measures the amount of work required to transform A into B. By minimising this distance we maximise the similarity between the two point sets. An alternative would be to minimise the maximum distance between a pair of matched points. However, the maximum distance is very sensitive to outliers. Consider, for example, the case where both the blue and the red point set contain a copy of the same regular grid and have one outlier each. The minimal (total) distance matching is the desired matching in which we correctly match the points from the grid, and match the outlier of A to the outlier of B. However, this matching is not optimal with respect to the maximal distance between a pair of matched points.

The remaining two criteria allow the user to navigate in the grid in the same way as in the map. This further improves the ease with which they can find a region.



We note that it may not be possible to completely satisfy a given criterion. This holds especially for the second and third criterion. For example when there are not enough grid points in a certain direction or when a region has more than four neighbours. Additionally, it may be the case the criteria themselves are contradictory; the distance criterion may require a point a to be matched to b_1, whereas the adjacencies criterion requires a to be matched to b_2.

We keep the red points stationary and allow two kinds of transformations on the blue point set: translation and (non-uniform) scaling. These transformations can greatly improve the quality of the matching. Note that we do not allow rotating the blue point set A. Rotating A by $90°$ simply corresponds to choosing a different grid. Other rotations of the input map may make it more difficult to find a given region. As a result it is also harder to locate a region in the more abstract grid representation.

Finally, there is one remaining issue: how to obtain a suitable grid? The number of regions and the outline of the map are important factors in determining which grid, or which set of grid cells, to use. We investigate this topic as well.

Results and Organisation. Our goal is to visualise geo-referenced data such that we can easily compare data for multiple regions and get a geographical overview as well. In Chapter 2 we consider several existing visualisation techniques and discuss their pros and cons with these goals in mind. Since we model our own visualisation technique as a point set matching problem we also give an overview of existing work in this field. We present our geographic grid embeddings technique in Chapter 3. The point set matching used in our technique has three criteria: distance, adjacencies, and directional relation. We treat computing an optimal matching with respect to each criterion separately. In the last part of Chapter 3 we discuss how to choose a suitable grid. We evaluate our technique on real world maps in Chapter 4 and close by discussing future work in Chapter 5.

2

Related Work

In this chapter we discuss existing literature related to geographic grid embeddings. In Section 2.1 we describe alternative methods for visualising geo-referenced data. The second part of this chapter focuses on solving point set matching problems. There is a vast amount of literature on this topic. We describe the most relevant approaches in Section 2.2.

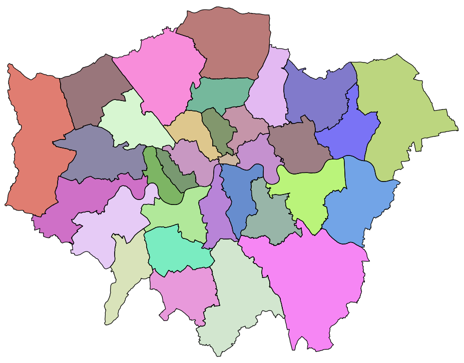
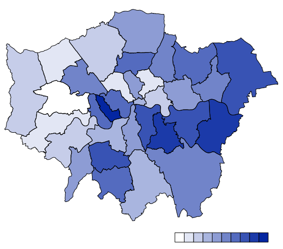
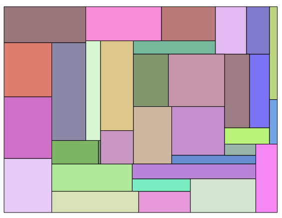


Figure 2.1: The map of the London boroughs that is used throughout this chapter.

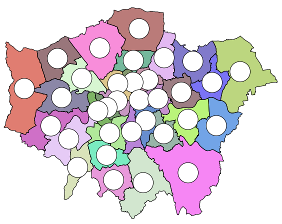
2.1 Visualising geo-referenced data



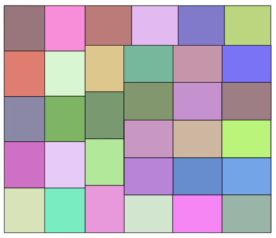
Perhaps the most well known visualisation for geo-referenced data is a *choropleth map*. In a choropleth map the regions are coloured proportional to the corresponding data. Choropleth maps are mainly suited to display scalar values, like the gross national product of a country or state sales tax rates [27]. We can display additional information by adding another *visual variable*. For example by using *lightness* (colour) for the first layer of information and *spacing* (texture) for the second. This may however impede the readability of the resulting map. The main disadvantage of choropleth maps is that it is difficult to compare data from different regions. The size of a region greatly influences how we interpret the data. Large regions are perceived as being important, whereas small regions are perceived unimportant, independent of the actual data. As a result choropleth maps are best used for scalar data on maps with roughly equal sized regions [27].



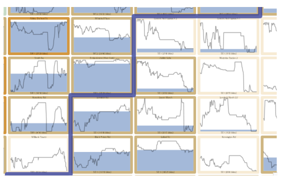
Alternatively, one can use an *(area) cartogram* to display scalar data [15]. In a cartogram the region's size is proportional to the data being displayed. This often means the shape of the regions becomes distorted. To keep the figure readable the region's shape is usually simplified (for example by representing each region by a rectangle, which yields a *rectangular cartogram* [25]). Because of this distortion it can become difficult to recognise and find a given region. Additionally, the different aspect ratios make it hard to compare the data for multiple regions.



A third option to display scalar data is to use a *proportional symbol map* [27]. In a proportional symbol map we place symbols, often geometric symbols such as a circles, on top of the regions in the map. The size of a symbol is depends on the data corresponding to the region. Van Kreveld, Schramm, and Wolff [30] study placing diagrams, for example pie- or bar charts, on maps. This makes their method suitable to display multi-variate data, and allows for easy comparison between data of different regions. However, both methods suffer from the same problem: adding the symbols or diagrams may result in a cluttered visualisation. This is especially true when the symbols or diagrams start to overlap, which easily happens when the symbols and diagrams are relatively large compared to their corresponding regions.



Our work is most closely related to *spatially ordered tree maps* [34]. A *tree map* is a recursive partition of a rectangle into rectangles [26]. Tree maps are designed to display hierarchical data, for example for visualising hard-disk usage [31]. Wood and Dykes [34] propose spatially ordered tree maps in which the rectangles are placed as closely to the geographic location of the region as possible while minimising the aspect ratio of the rectangles. This results in a visualisation in which each region is represented by an almost square rectangle. These rectangles can then be used to display the data corresponding to the region, similar to our grid cells. This makes the spatially ordered tree maps suited to display both scalar and multi-variate data.



Because not all rectangles in a tree map have to have the same size, the resulting visualisation does not have to be a regular grid. Wood and Dykes use (a variation of) their spatially ordered tree maps in the London bike grid [33] that does yield a regular structure. This system visualises the availability of bicycles in each docking station of Barclays Cycle Hire system. One of the more advanced features in this visualisation is that it preserves the relative position of the docking stations compared to the river Thames. This approach essentially yields a geographic grid embedding, and inspired our study. The main difference between their method and our own approach is that we aim for a mapping of regions to the grid which has a provable quality with respect to our criteria.

2.2 Solving point set matching problems

We model our geographic grid embeddings as a (one-to-one) point set matching problem. In this section we discuss existing methods for solving such point set matching problems. Point set matching problems have been extensively studied in the literature. A survey of existing methods is presented by Alt and Guibas [1], or more recently by Veltkamp and Hagedoorn [32].

Point matching problems usually occur in the context of image comparisons. The similarity of two images is determined by computing the distance between a set of points from one image and a set of points from another image. This often involves comparing sets of points with different sizes. In these cases it is common to express the distance using the Hausdorff distance with the $L_p(a,b) = (|(a_x - b_x)^p| + |(a_y - b_y)^p|)^{1/p}$, $1 \leq p \leq \infty$, distance, or the L_2^2 squared Euclidean distance as the underlying metric [21].

Huttenlocher, Kedem, and Sharir [20] provide an algorithm to compute the Hausdorff distance using the upper envelope of Voronoi surfaces. Given two sets A and B of points in the plane they can compute the minimum Hausdorff distance between A and B, and the translation of A that achieves this minimum, in $O(nm(n+m)\log nm)$ time, where n is the size of A and m is the size of B.

Hagedoorn and Veltkamp [17] give an approximation algorithm for minimising the Hausdorff distance, and a new distance they call *absolute distance*, under affine transformations. Their approach subdivides the transformation space to search for a set of suitable transformations. They show that if they have a set of transformations that does not yield a matching that approximates the minimal Hausdorff distance within ϵ they can find a "smaller" set that does.

None of these approaches is guaranteed to yield one-to-one matchings. Atkinson [4] presents an algorithm for computing a one-to-one matching in case the input point sets are assumed to be copies the same point set, possibly with affine transformations applied on them. We can obtain this so called *exact matching* in $O(n \log n)$ time by computing the polar coordinates of all points in a set A with the centroid of A as origin. This yields an ordering of the points. The same procedure is used for input set B, after which the matching can be computed by checking if the ordering of B, O_B, is a cyclic shift of the ordering of A, O_A. This can be done by checking if O_B is a substring of $O_A +\!\!+ O_A$ using a fast string matching algorithm.

Hong and Tan [19] present a similar procedure. They also transform the coordinates of the points in A to polar coordinates around the centroid. They call this the *canonical form* of A. After computing the canonical forms of both A and B they find the rotation that aligns them to compute a matching. The transformation step takes linear time, and they compute the rotation in $O(n^2)$ average time. The algorithm by Hong and Tan can also be used when the points sets are not exact copies of each other. They allow a point p to be matched to q if q lies in the *error area* of p. The error area $E(p)$ of a point p is a convex polygon for which the distance between p and the closest point on the boundary of $E(p)$ and the distance between p and the furthest point on the boundary differs by at most a constant factor. It is assumed that p is contained in $E(p)$, and that for any point q we can check if $E(p)$ contains q in constant time. Furthermore, all error regions have to be pairwise disjoint. Sprinzak and Werman [28] extend Hong and Tan's method for point sets in arbitrary dimensions.

Alt et al. [3] present algorithms for several types of point set matching problems. Their algorithms can handle both the L_2 Euclidian distance metric, and the L_∞ maximum distance metric. In case all points have disjoint error areas with radius ϵ (a disk with radius ϵ in the Euclidean case and a square with side lengths 2ϵ in case of the maximum distance metric) they present an $O(n \log n)$ algorithm to compute the minimal distance matching, and the translation that yields this matching.

In case the error regions are not given, but instead we should decide whether or not there is a translation with corresponding matching such that the distance between a pair of matched points is at most ϵ, Alt et al. present an $O(n^6)$ algorithm. When we actually want to find the smallest such ϵ the running time increases to $O(n^6 \log n)$. Finally, Alt et al. [3] show that if we want to allow rotating and mirroring our point sets as well we can solve the decision version of the problem in $O(n^8)$ time.

Efrat and Itai [13] investigate *bottleneck matching.* They show how to find a one-to-one matching and translation that minimises the maximum distance between the pairs of points (the distance using the L_∞ metric) in $O(n^5 \log^2 n)$ time. This improves the results of Alt et al. [3] by almost a factor n. For the decision version of the problem their algorithm runs in $O(n^5 \log n)$ time.

There is also a point set matching approach by Cohen and Guibas [10] which uses the *Earth Mover's Distance.* In this setting each point has a certain weight. The amount of work to match a point $a \in A$ with a point $b \in B$ is determined by the distance between a and b and the weight of a that is matched to b. The earth mover's distance expresses the minimal work required to match A and B. This approach allows for complete matchings as well as *partial matches* in which we only match a part of the weight. Cohen and Guibas [10] present an algorithm that can find a transformation of point set A and a matching that locally minimises the earth mover's distance. In the case of equal weight point sets and L_2^2 as the underlying distance metric they show how to obtain the matching with the global minimum distance. This is one of the methods that we use in the computation of our geographic grid embeddings.

Recently Alt, Scharf, and Schymura [2] proposed a probabilistic method for matching planar regions. Their algorithm picks a random set of points A in one region and a random set of points B in the other. It then computes a transformation (consisting of a translation and a rotation) such that the area of overlap between the planar regions is close to maximal. They argue that it may be possible to extend their approach to compute a matching under affine transformations.

3 Computing a Geographic Grid Embedding

In this chapter we introduce geographic grid embeddings. A geographic grid embedding matches regions to cells in a regular grid. We have three criteria for determining the quality of such a matching: distance, adjacencies, and directional relation. We investigate optimising the matching for the distance criterion in Section 3.1. In Section 3.2 we focus on the adjacencies criterion, and in Section 3.3 we discuss the directional relation criterion. Finally, in Section 3.4 we consider the problem of choosing a suitable grid.

3.1 Minimising the distance

We start by formalising the distance criteria. We first introduce some notation. For a point $a = (a_x, a_y)$ and a translation $t = (t_x, t_y)$ we write $a + t = (a_x + t_x, a_y + t_y)$. Similarly, for a scaling $\lambda = (\lambda_x, \lambda_y)$ we write $\lambda a = (\lambda_x \cdot a_x, \lambda_y \cdot a_y)$. We denote a transformation (either translation or scaling) for which both components have the same value c with $\overline{c} = (c, c)$.

Let $\phi : A \to B$ be a one-to-one matching for the point sets A and B, let t be a translation and let λ be a scaling. Then we define the *total distance* of matching ϕ with translation t and scaling λ as

$$D(\phi, t, \lambda) = \sum_{a \in A} d(\lambda a + t, \phi(a))$$

where, d is the underlying distance metric. Additionally, we define some shorthands: $D_{\mathcal{T}}(\phi, t) = D(\phi, t, \overline{1})$, $D_{\Lambda}(\phi, \lambda) = D(\phi, \overline{0}, \lambda)$ and $D_I(\phi) = D(\phi, \overline{0}, \overline{1})$.

We now want to find a matching together with a translation and/or scaling that minimises the total distance. More formally, let Φ be the universe of one-to-one matchings between A and B, let $\mathcal{T}$ be the universe of translations, and let Λ be the universe of scalings, then we try to find a matching $\phi^* \in \Phi$, a translation $t^* \in \mathcal{T}$, and a scaling $\lambda^* \in \Lambda$ such that

$$D(\phi^*, t^*, \lambda^*) = \min_{\phi \in \Phi, t \in \mathcal{T}, \lambda \in \Lambda} D(\phi, t, \lambda).$$

We start by solving an easier variant of the problem in which we do not allow translation or scaling. This means we want a matching ϕ^* that minimises D_I. We discuss how to compute such a matching in Section 3.1.1. We use this as a building block for the more difficult versions of the problem. In Section 3.1.2 we investigate these more difficult versions using the L_1 Manhattan distance as the underlying metric, and in Section 3.1.3 we do the same for the case of the L_2^2 squared Euclidean distance.

3.1.1 Computing point set matchings

To compute a minimal distance matching for two sets of points A and B we use the same approach as Cohen [9]. This means we consider the problem as an instance of the *transportation problem*, which we model and solve using linear programming.

We say each point $a \in A$ has a supply of one, and each point $b \in B$ has a demand of one. Each point $a \in A$ can supply exactly one point $b \in B$ for a cost of $d(a, b)$. We model this by variable f_{ab} denoting the supply, or *flow*, from a to b. The objective is to find an assignment of the flow that minimises the weighted total cost. This corresponds to minimising D_I. We can express this in the following linear program:

$$\text{minimize} \sum_{a \in A} \sum_{b \in B} f_{ab} d(a, b)$$

subject to:

$$\begin{aligned} \sum_{b \in B} f_{ab} &= 1 && \forall a \in A \\ \sum_{a \in A} f_{ab} &= 1 && \forall b \in B \\ 0 \leq f_{ab} &\leq 1 && \forall a \in A, b \in B \end{aligned}$$

Since all supplies and demands have integer values it can be shown that all variables in the optimal flow f_{ab} also have integer values [18]. Therefore f_{ab} represents an optimal one to one matching $\phi \in \Phi$ that matches a to b if and only if $f_{ab} = 1$.

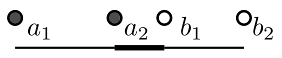
To solve this linear program we can use a regular LP solver. However, there are also specialised methods, like the *transportation simplex* *method*, that use the structure in the transportation problem [18] to reduce computation times. Our linear program is even more specific which means we can use algorithms for the *assignment problem* like the *Hungarian method* [23], the *auction algorithm* [6], or Edmond's alternating path algorithm [12]. This leads to the following theorem:

Theorem 1 *Given two sets A and B of n points in the plane, a one-to-one matching ϕ that minimises D_I can be computed in $O(n^3)$ time.*

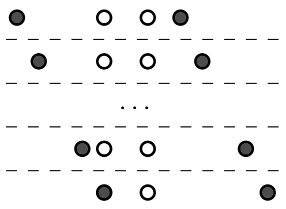
The problem of finding a minimal distance matching under translation and/or scaling is significantly more difficult than finding a matching that minimises D_I. However, we can use this method as a building block for solving these more difficult versions of the problem. This follows from the observation that finding a minimal distance matching between A and B for a given translation t and scaling λ is identical to finding a minimal distance matching between $A' = \{\lambda a + t \mid a \in A\}$ and B. In case the translation and scaling are not given we can now focus on finding a suitable set of translations $T \subset \mathcal{T}$ and a suitable set of scalings $S \subset \Lambda$ for which we can use this procedure to find a matching that minimises $D_{\mathcal{T}}$ or D_{Λ}.

3.1.2 Using the L_1 Manhattan distance

In this section we focus on finding a minimal distance matching when using the L_1 Manhattan distance as the underlying metric. In other words: we use $d(a, b) = L_1(a, b) = |a_x - b_x| + |a_y - b_y|$ in our total distance D. We note that in general there is no unique matching that minimises D, or even D_I. If we consider four points a_1, a_2, b_1, b_2 on a line, then there are two possible matchings: ϕ which matches a_i to b_i ($i \in \{1, 2\}$) and ψ which matches a_1 to b_2 and a_2 to b_1. Both matchings have the same total distance, which means both are optimal with respect to D_I.



A similar observation holds for finding a suitable translation and/or scaling. In general there is no unique translation (scaling) with corresponding matching that minimises $D_{\mathcal{T}}$ (D_{Λ}). It is very well possible that there are multiple translations $t_1, .., t_k$ with corresponding minimal distance matchings $\phi_1, .., \phi_k$ that all have the same total distance $D_{\mathcal{T}}(\phi_1, t_1) = ... = D_{\mathcal{T}}(\phi_k, t_k) = c$. In fact, there can be infinitely many such translations. The same is true for scalings.



Preliminaries. Before we show how to compute a minimal distance matching under translation we prove the following useful lemmas:

Lemma 2 *Let f be a continuous piecewise linear function in which the leftmost segment is decreasing and the rightmost segment is increasing, and let $m = f(x_m) = \min_x f(x)$ be the minimum value f can attain. Then there is a breakpoint v for which $f(v) = m$.*

Proof. Let s be the range of a segment of f on which m occurs, and let g be the function that defines this segment. So for all $x \in s$ we have $f(x) = g(x) = ax + b$ for some real values a and b. We now prove the lemma using a case distinction on a:

The case $a = 0$ is trivial; let v be one of the endpoints of s, and hence one of the breakpoints of f. Since $a = 0$ we have $f(v) = g(v) = g(x_m) = f(x_m) = m$. We prove the case $a < 0$ by contradiction. Assume x_m is not the rightmost point of s. Let $x' \in s$ be a point to the right of x_m, i.e. $x' > x_m$. This gives the contradiction $f(x') = g(x') < g(x_m) = f(x_m) = m$. Hence x_m is the rightmost point of s, and therefore a breakpoint of f. The remaining case, $a > 0$, is symmetrical to $a < 0$. This completes the proof. $\square$

Lemma 3 *Let f be a convex function. If f is minimal at x_1 and x_2, $x_1 \leq x_2$, then f is minimal at any x for which $x_1 \leq x \leq x_2$.*

Proof. In case $x_1 = x_2$ the lemma trivially holds so we focus on the case $x_1 < x_2$. We prove this by contradiction: assume that there is an x, $x_1 < x < x_2$, for which $f(x) > f(x_1)$. Then there is an α, $0 < \alpha < 1$, such that $x = (1-\alpha)x_1 + \alpha x_2$.

Since f is convex we have $f(x) \leq (1-\alpha)f(x_1) + \alpha f(x_2)$. Because f is minimal for both x_1 and x_2 it follows that $f(x_2) = f(x_1)$. This gives us $f(x) \leq (1-\alpha)f(x_1) + \alpha f(x_1) = f(x_1)$ which contradicts $f(x) > f(x_1)$. The lemma follows. $\square$

Minimal distance matchings under translation. To find a minimal distance matching under translation, i.e. a matching that minimises $D_{\mathcal{T}}$, we identify a (finite) set of translations $T \subset \mathcal{T}$ such that there is a translation $t \in T$ that allows for a minimal distance matching. We then use the algorithm from Section 3.1.1 for each translation in T to find an optimal matching.

We say a translation t is a *horizontal translation* if and only if $t = (c, 0)$ for some $c \in \mathbb{R}$. Two point sets A and B are *x-aligned* if (and only if) there is a point $a \in A$ and a point $b \in B$ with $a_x = b_x$. We define *vertical translation* and *y-aligned* symmetrically.

The following observation is going to help us identify a finite set of translations T. For any matching between two point sets A and B that are not x-aligned we can decrease the distance of the matching by x-aligning A and B. This is depicted in Figure 3.1. To prove this we decompose the total distance $D_{\mathcal{T}}(\phi, t)$ into two parts: $D_{\mathcal{T}}(\phi, t) = X_{\mathcal{T}}(\phi, t) + Y_{\mathcal{T}}(\phi, t)$, with $X_{\mathcal{T}}(\phi, t) = \sum_{i=1}^{n} |a_x + t_x - b_x|$, and $Y_{\mathcal{T}}(\phi, t) = |a_y + t_y - b_y|$.

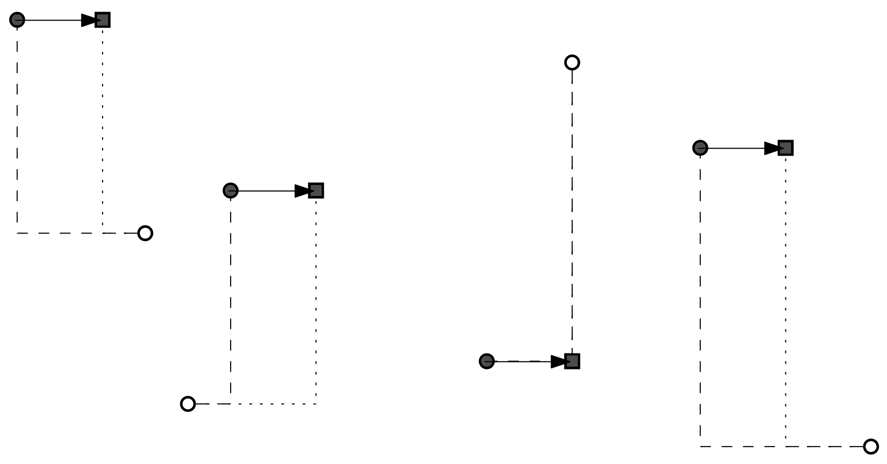


Figure 3.1: We can improve a matching between A and B (indicated by the dashed lines) by x-aligning the point sets (the dotted lines).

Furthermore, we again introduce shorthands $X_I(\phi) = X_{\mathcal{T}}(\phi, \overline{0})$ and $Y_I(\phi) = Y_{\mathcal{T}}(\phi, \overline{0})$. Given a one-to-one matching ϕ we write $X'(t) = X_{\mathcal{T}}(\phi, t)$. This function X' has the following properties:

1. X' is the sum of the set of continuous convex piecewise linear functions $\mathcal{F}_x = \{f \mid a \in A \wedge f(t) = |a_x + t_x - \phi(a)_x|\}$,
2. $X'(\overline{0}) = X_I(\phi)$,
3. X' is a piecewise linear function with as breakpoints the union of all breakpoints of functions in $\mathcal{F}_x$,
4. X' is continuous, and
5. X' is convex.

The first two properties follow directly from the definition of X'. The remaining properties result from the fact that X' is composed from functions of $\mathcal{F}_x$ together with some basic function calculus (e.g. the sum of two linear functions is linear). We also define the function $Y'(t) = Y_{\mathcal{T}}(\phi, t)$, which has similar properties. We now prove the following lemmas:

Lemma 4 *Let A and B be two non x-aligned sets of n points in the plane, and let ϕ be a one-to-one matching between A and B. Then there is a horizontal translation t^* such that $A^* = \{a + t^* \mid a \in A\}$ and B are x-aligned and $D_{\mathcal{T}}(\phi, t^*) \leq D_I(\phi)$.*

Proof. Since we are searching for a horizontal translation t^* it follows that $Y_{\mathcal{T}}(\phi, t^*) = Y_I(\phi)$. This means that we need to prove there is a t^* such that A^* and B are x-aligned and $X_{\mathcal{T}}(\phi, t^*) \leq X_I(\phi)$. We show this using the function X'.

Given the matching ϕ the function X' is a convex continuous piecewise linear function with as breakpoints the union of all breakpoints of functions in $\mathcal{F}_x$. Lemma 2 then gives us X' attains its minimum at one of these breakpoints.

All functions in $\mathcal{F}_x$ are of the form $f(t) = |a_x + t_x - \phi(a)_x|$ so they have exactly one breakpoint: the point in which $f(t)$ attains its minimal value 0. It follows that if X' attains its minimal value $X'(t^*)$ at breakpoint t^* there is a function $f \in \mathcal{F}_x$ which is also minimal at t^*. Since $f(t^*) = 0$ there is a point $a \in A$ for which $a + t^*$ and $\phi(a)$ x-align. Hence A^* and B are x-aligned.

Using that $X'(t^*)$ is minimal it follows that $X_{\mathcal{T}}(\phi, t^*) = X'(t^*) \leq X'(\overline{0}) = X_I(\phi)$, and therefore also $D_{\mathcal{T}}(\phi, t^*) \leq D_I(\phi)$. We conclude that A^* and B are x-aligned and that $D_{\mathcal{T}}(\phi, t^*) \leq D_I(\phi)$. This completes the proof. □

Lemma 5 *Let A and B be two non y-aligned sets of n points in the plane, and let ϕ be a one-to-one matching between A and B. Then there is a vertical translation t^* such that $A^* = \{a + t^* \mid a \in A\}$ and B are y-aligned and $D_{\mathcal{T}}(\phi, t^*) \leq D_I(\phi)$.*

Proof. Symmetrical to Lemma 4. □

We now consider the set of translations T such that $t \in T$ if and only if t both x-aligns and y-aligns A and B. A translation $t \in T$ x-aligns a pair of points (a_x, b_x), and independently y-aligns a pair of points (a_y, b_y). This means T contains at most n^4 translations. From Lemmas 4 and 5 it follows that T contains a translation t that allows for a minimal distance matching. We then use the algorithm from Section 3.1.1 to compute a matching between $A_t = \{a + t \mid a \in A\}$ and B for all $t \in T$, and pick a matching with the smallest distance. This matching minimises $D_{\mathcal{T}}$. By combining this result with Theorem 1 we obtain a method to compute a minimal distance matching under translation in $O(n^4 \cdot n^3) = O(n^7)$ time. If we use the algorithm of Vaidya [29] instead we can improve this to $O(n^4 \cdot n^2 (\log n)^3) = O(n^6 (\log n)^3)$. This leads to the following theorem:

Theorem 6 *Given two sets A and B of n points in the plane, a one-to-one matching ϕ and translation t that minimise $D_{\mathcal{T}}$ can be computed in $O(n^6 (\log n)^3)$ time.*

We note that in the case of our geographic grid embeddings we can use the structure of point set B to improve the running time. Since B is a regular grid any two points b_1, b_2 in the same column have the same x-coordinate. Hence x-aligning a point $a \in A$ with b_1 has the same effect as x-aligning a with b_2. The same holds for any two points in the same row. This implies the following corollary:

Corollary 7 *Given a set A of n points in the plane and a set B of n grid points in an $R \times C$ grid, a one-to-one matching ϕ and translation t that minimise $D_{\mathcal{T}}$ can be computed in $O(nCnR \cdot n^2(\log n)^3) = O(n^5(\log n)^3)$ time.*

Additionally, we can characterise *all* translations that have a corresponding matching that minimise $D_{\mathcal{T}}$. We consider the universe of translations $\mathcal{T}$ as a plane, which we call the *translation plane.* On the horizontal axis we have the x-coordinate of the translations and on the vertical axis the y-coordinate. We represent a translation as a point in this translation plane. We observe that there is a rectangle R in this translation plane such that all translations in R have a corresponding matching that minimises $D_{\mathcal{T}}$. This is depicted in Figure 3.2.

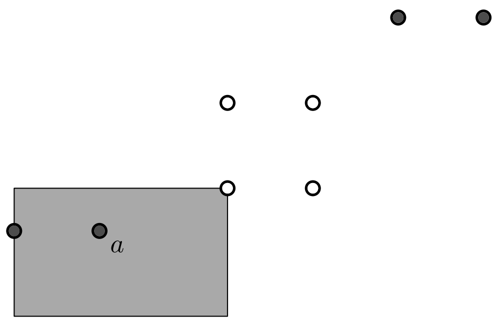


Figure 3.2: The optimal translations for a point a form a rectangle in the translation space.

This leads us to the following lemma, in which we use $D'(t) = D_{\mathcal{T}}(\phi, t)$ for a given matching ϕ.

Lemma 8 *Let A and B be two sets of n points in the plane, and let ϕ be a one-to-one matching between A and B. The set of translations T^* that minimises D' forms a rectangle R in the translation plane.*

Proof. We have to show that there is a rectangle R such that all translations in R are optimal, and that these translations are the only ones that are optimal. We start with the first part.

We again consider the decomposition of $D_{\mathcal{T}}$ (D') into $X_{\mathcal{T}}$ and $Y_{\mathcal{T}}$ (X' and Y'). Since X' is a convex function it follows from Lemma 3 that there is exactly one range of values, say $[x_\ell, x_u]$, for which X' is minimal. Hence any translation t such that $x_\ell \leq t_x \leq x_u$ minimises X'. A symmetric argument gives us a single range $[y_\ell, y_u]$ such that any translation t with $y_\ell \leq t_y \leq y_u$ minimises Y'.

Since X' and Y' are independent this means all translations in the rectangle $R = [x_\ell, x_u] \times [y_\ell, y_u]$ minimise both X' and Y'. Hence all translations in R also minimise D'.

What remains to prove is that all optimal translations lie inside R. Assume t is an optimal translation that lies outside of $R = [x_\ell, x_u] \times [y_\ell, y_u]$, and that R is maximal (i.e. there is no rectangle $R' \supset R$ such that all translations in R' are maximal). Consider the case in which $t_x \notin [x_\ell, x_u]$ (the case that $t_y \notin [y_\ell, y_u]$ is symmetrical). We now show that for translation $t' = (x_\ell, t_y)$ we have $D'(t') < D'(t)$. Since $t'_y = t_y$ we have $Y'(t') = Y'(t)$. This means we should show $X'(t') < X'(t)$.

All translations t^* with $t^*_x \in [x_\ell, x_r]$ minimise X'. This gives us $X'(t') \leq X(t)$. Using that X' is convex and applying Lemma 3 we obtain that there is a single range for which X' is minimal. This together with the fact that R is maximal, and $t_x \notin [x_\ell, x_u]$ gives us $X'(t') < X(t)$. We therefore have $D'(t') < D'(t)$, which contradicts our assumption that t was optimal. Hence we conclude all optimal translations lie within R. This completes the proof. $\square$

Minimal distance matchings under scaling. For scaling we use the same procedure as for translation. We start by defining an *horizontal scaling* $\lambda = (c, 0)$, and we again use the decomposition of D_Λ into separate components for both x and y. For scaling this gives us $D_\Lambda(\phi, \lambda) = X_\Lambda(\phi, \lambda) + Y_S(\phi, \lambda)$ with $X_\Lambda(\phi, \lambda) = \sum_{a \in A} |\lambda_x a_x - \phi(a)_x|$ and Y_S defined similarly.

Given a matching ϕ we define $X''(\lambda) = X_\Lambda(\phi, \lambda)$. This function X'' then has the following properties:

1. X'' is the sum of the set of continuous convex piecewise linear functions $\mathcal{G}_x = \{g \mid a \in A \wedge g(\lambda) = |\lambda_x a_x - \phi(a)_x|\}$,
2. $X''((1, 1)) = X_I(\phi)$,
3. X'' is a piecewise linear function with as breakpoints the union of all breakpoints of functions in $\mathcal{G}_x$,
4. X'' is continuous, and
5. X'' is convex.

Similarly we define a *vertical scaling* and the function Y''. Using the exact same arguments as in Lemma 4 we obtain the following two lemmas:

Lemma 9 *Let A and B be two non x-aligned sets of n points in the plane, and let ϕ be a one-to-one matching from A to B. Then there is a horizontal scaling λ^* such that $A^* = \{\lambda^* a \mid a \in A\}$ and B are x-aligned and $D_\Lambda(\phi, \lambda^*) \leq D_I(\phi)$.*

Lemma 10 *Let A and B be two non x-aligned sets of n points in the plane, and let ϕ be a one-to-one matching from A to B. Then there is a vertical scaling λ^* such that $A^* = \{\lambda^* a \mid a \in A\}$ and B are x-aligned and $D_\Lambda(\phi, \lambda^*) \leq D_I(\phi)$.*

Using these lemmas we obtain a set S of at most n^4 scalings such that there is a $\lambda \in S$ that allows for a minimal distance matching. This implies Theorem 11. We again use that x-aligning (y-aligning) a point a with either point b_1 or b_2 from the same column (row) yields the same scaling. This yields Corollary 12.

Theorem 11 *Given two sets A and B of n points in the plane, a one-to-one matching ϕ and scaling λ that minimise D_Λ can be computed in $O(n^6(\log n)^3)$ time.*

Corollary 12 *Given a set A of n points in the plane and a set B of n grid points in an $R \times C$ grid, a one-to-one matching ϕ and scaling λ that minimise D_Λ can be computed in $O(nCnR \cdot n^2(\log n)^3) = O(n^5(\log n)^3)$ time.*

Minimal distance matchings under scaling and translation. We also investigate the version of the problem in which we allow both scaling and translating the point set A. The idea is to generalise the approach used for only translation or scaling: if one transformation can x-align (y-align) one pair of points, two transformations can x-align (y-align) two pairs of points. Suppose that the translation t should x-align a and b, and the scaling λ should x-align a' and a'. This gives us the system of equations

$$\begin{aligned} t_x &= |b_x - a_x| \\ \lambda_x &= |b'_x - a'_x|. \end{aligned}$$

This system of equations has a unique solution only if a_x, a'_x, b_x, and b'_x are independent. Unfortunately this does not necessarily hold in our setting. As a result, the set of transformations that x-aligns two points can still contain infinitely many transformations. This means that at this point give only a heuristic to minimise D.

We can, for example, consider the following iterative approach similar to the FT-iteration scheme of Cohen and Guibas [10]. We first compute a matching that minimises $D_\mathcal{T}$ and then use the translated point set in the computation of a minimal distance matching under scaling. We repeat this procedure until the total distance no longer changes or we reach a threshold on the number of iterations. In the first case we arrive at a local minimum.

Translation spaces and the L_1 min-max distance. To get a unique minimal distance matching with the L_1 distance we can consider not only minimising the sum of the distances between matched points, but also the maximal distance between a pair of matched points. We refer to this as the L_1 *min-max distance.*

More specifically, we consider the set of matching-translation pairs $\Psi \times T' \subseteq \Phi \times \mathcal{T}$ such that for all $(\psi, t') \in \Psi \times T'$, we have $D_{\mathcal{T}}(\psi, t') = \min_{\phi \in \Phi, t \in \mathcal{T}} D_{\mathcal{T}}(\phi, t)$. We now choose the matching $\psi \in \Psi$ that (lexicographically) minimises the maximal distance $m(a) = \max_{a \in A} d(a, \phi(a))$.

We observe an interesting phenomenon in the set T of translations that allow for the matching that minimises the L_1 min-max distance. We identify the following four cases for the set T, these are depicted in Figure 3.3:

- T contains a single translation that allows for a minimal distance matching. This can be represented as a single point in the translation plane (Figure 3.3a),
- T is a set of translations that lie on a diagonal line in the translation plane (Figure 3.3b),
- T consists of two separate translations with the same x-component, say $t_1 = (x, y_1)$ and $t_2 = (x, y_2)$, with $y_2 \leq y_1$. So any translation $t' = (x, y')$ with $y_2 \leq y' \leq y_1$ does not allow for a minimal distance matching. This yields two separate x-aligned points in the translation plane. Figure 3.3c_1 corresponds to t_1 and Figure 3.3c_2 corresponds to t_2.
- two separate translations with the same y-component. This is the previous case rotated by 90° (Figure 3.3d).

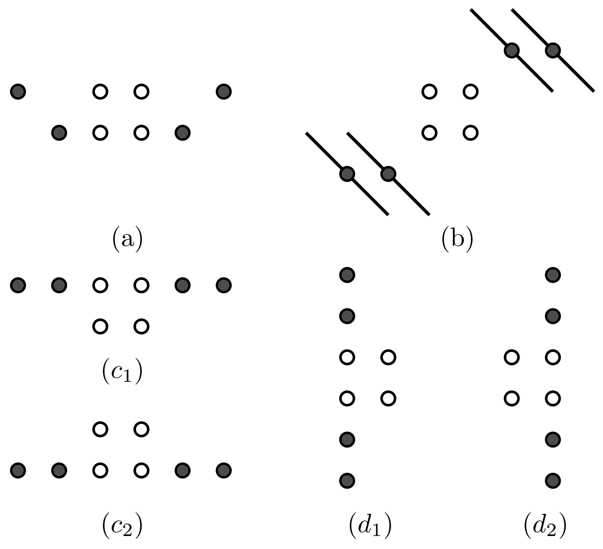
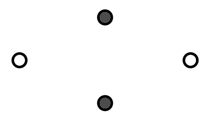


Figure 3.3: Optimal translations for L_1 min-max in the translation plane. (a) a single point (representing the translation $\bar{0}$). (b) a diagonal line-segment. (c) two separate x-aligned points. (d) two separate y-aligned points.

3.1.3 Using the L_2^2 Squared Euclidean distance



In the case in which we use the L_2^2 squared Euclidean distance in our total distance D there is still no unique minimal distance matching. However, finding the minimal distance matching under translation is easier than in the case of the L_1 distance. Cohen and Guibas [10] sketch a proof that shows that there is a unique translation that allows for a minimal distance matching: the translation that aligns the centroids of the point sets. For completeness we include the full proof.

Theorem 13 *Let A and B be two sets of n points in the plane, and let t^* be the translation that aligns the centroids of A and B. There exists a matching ϕ^* such that $D_{\mathcal{T}}(\phi^*, t^*) = \min_{\phi\in\Phi, t\in\mathcal{T}} D_{\mathcal{T}}(\phi, t)$.*

Proof. We use the same formulation of the total distance as is used in Section 3.1.1. For a given matching ϕ we consider the flow f_{ab} between a and b that is one if ϕ matches a to b, and zero otherwise. For a translation t we now define $D'(t)$ expressing the total distance $D_{\mathcal{T}}(\phi, t)$ as a function of t:

$$D'(t) = \sum_{a\in A}\sum_{b\in B} f_{ab} d(a+t, b) = \sum_{a\in A}\sum_{b\in B} f_{ab}((a_x+t_x-b_x)^2+(a_y+t_y-b_y)^2)$$

To find the translation that minimises D' we now simply compute the extrema of D'. To this end we compute the derivative of D' with respect to t. We start with the x-component:

$$\begin{aligned}
\frac{\partial}{\partial t_x} D'(t) &= \sum_{a\in A}\sum_{b\in B} f_{ab} \frac{\partial}{\partial t_x}((a_x + t_x - b_x)^2 + (a_y + t_y - b_y)^2) \\
&= \sum_{a\in A}\sum_{b\in B} f_{ab}(\frac{\partial}{\partial t_x}(a_x + t_x - b_x)^2 + 0) \\
&= \sum_{a\in A}\sum_{b\in B} f_{ab} 2(a_x + t_x - b_x)
\end{aligned}$$

By solving the equation $\frac{\partial}{\partial t_x} D'(t) = 0$ we get the extremal points of D':

$$\begin{aligned}
\sum_{a\in A}\sum_{b\in B} f_{ab}2(a_x+t_x-b_x) &= 0\\
\sum_{a\in A}\sum_{b\in B} f_{ab}t_x + \sum_{a\in A}\sum_{b\in B} f_{ab}(a_x-b_x) &= 0\\
t_x\sum_{a\in A}\sum_{b\in B} f_{ab} &= -\sum_{a\in A}\sum_{b\in B} f_{ab}(a_x-b_x)\\
t_x &= \frac{\sum_{a\in A}\sum_{b\in B} f_{ab}(b_x-a_x)}{\sum_{a\in A}\sum_{b\in B} f_{ab}}
\end{aligned}$$

Since the flow models a one-to-one matching we can use that $\sum_{a\in A}\sum_{b\in B} f_{ab} = n$, that $\sum_{b\in B} f_{ab} = 1$ for a given a, and $\sum_{a\in A} f_{ab} = 1$ for a given b. Hence we obtain:

$$\begin{aligned}
t_x &= \frac{\sum_{a\in A}\sum_{b\in B} f_{ab}(b_x-a_x)}{\sum_{a\in A}\sum_{b\in B} f_{ab}}\\
&= \frac{\sum_{a\in A}\sum_{b\in B} f_{ab}b_x - \sum_{a\in A}\sum_{b\in B} f_{ab}a_x}{n}\\
&= \frac{\sum_{b\in B} b_x}{n} - \frac{\sum_{a\in A} a_x}{n}
\end{aligned}$$

Using a similar derivation we obtain $t_y = \frac{\sum_{b\in B} b_y}{n} - \frac{\sum_{a\in A} a_y}{n}$. Hence the translation t^* that minimises D' aligns the centroids of A and B. Since $D'(t) = D_{\mathcal{T}}(\phi, t)$ this means t^* satisfies $D(\phi, t^*) = \min_{t\in\mathcal{T}} D_{\mathcal{T}}(\phi, t)$. This holds for an arbitrary matching ϕ, and therefore also for an optimal matching ϕ^*. The lemma follows. $\square$

The Squared Euclidean distance and scaling. We investigate using the same approach that we used for translation for scaling as well. Let f_{ab} again denote the flow from a to b for a given matching ϕ. We now consider the total distance D_Λ as a function D'' of a scaling λ. This gives us the following definition of D'':

$$D''(\lambda) = \sum_{a\in A}\sum_{b\in B} f_{ab}d(\lambda a, b) = \sum_{a\in A}\sum_{b\in B} f_{ab}((\lambda_x a_x - b_x)^2 + (\lambda_y a_y - b_y)^2)$$

We again compute the derivative of D'' with respect to λ. For the x-component we obtain

$$\frac{\partial}{\partial\lambda_x}D''(\lambda) = \sum_{a\in A}\sum_{b\in B} f_{ab}2(\lambda_x a_x^2 - a_x b_x).$$

However solving the equation $\frac{\partial}{\partial \lambda_x} D''(\lambda) = 0$ does not yield a unique solution for λ_x. We obtain the equation

$$\lambda_x = \frac{\sum_{a \in A} \sum_{b \in B} f_{ab} a_x b_x}{\sum_{a \in A} a_x^2}$$

which we cannot simplify further. What we can do is the following: let a^- be the point with the smallest x-coordinate, and a^+ is the point with the largest x-coordinate. This yields the following bounds on λ_x:

$$\frac{b_x^- \sum_{a \in A} a_x}{\sum_{a \in A} a_x^2} \leq \lambda_x \leq \frac{b_x^+ \sum_{a \in A} a_x}{\sum_{a \in A} a_x^2}$$

Unfortunately this still leaves a search space containing infinitely many scalings. We currently have no way of reducing this further. The same is true in the case we allow both translation and scaling.

3.2 Preserving adjacencies

The second criterion for the matching in our geographic grid embeddings is that it preserves the adjacencies between neighbouring regions. We consider the dual graph $G = (A, E)$ with an edge $(a, a') \in E$ if and only if the regions represented by a and a' are neighbouring. Similarly, we represent the grid by its dual graph $H = (B, Z)$. We then try to find the matching that preserves the maximum number of adjacencies from G. This means our problem is an instance of the *maximal common (edge) subgraph* problem.

The maximal common subgraph problem. Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the maximal common edge subgraph problem is to find the maximal size subsets $E_1' \subseteq E_1$ and $E_2' \subseteq E_2$ such that (V_1, E_1') and (V_2, E_2') are isomorphic. Garey and Johnson [14] prove that the corresponding decision problem for general graphs is NP-complete by reduction from CLIQUE. Since our version of the problem does not involve general graphs but a planar graph and a grid graph this does not necessarily mean our problem is also NP-hard. However, we now show that it is NP-complete to embed a planar graph into a grid graph while preserving at least k adjacencies. More formally, we prove the ADJACENCY PRESERVING GRID EMBEDDING problem as defined as follows is NP-complete:

ADJACENCY PRESERVING GRID EMBEDDING.

Given a planar graph $G = (V, E)$ and a grid graph $H = (N, Z)$ with $|V| = |N|$, is it possible to find an embedding $\phi : G \hookrightarrow H$ that preserves at least k adjacencies from G?

Clearly Adjacency Preserving Grid Embedding is the decision version of the problem we encounter in our geographic grid embeddings:

Maximum Adjacency Preserving Grid Embedding.

Given a planar graph $G = (V, E)$ and a grid graph $H = (N, Z)$ with $|V| = |N|$, find an embedding $\phi : G \hookrightarrow H$ that maximises the number of preserved adjacencies from G.

We show Adjacency Preserving Grid Embedding is NP-complete by reduction from 3-Partition:

3-Partition.

Given a natural number w, a multiset X of $3n$ natural numbers such that $w/4 < x < w/2$ for all $x \in X$, is possible to partition X into n multisets $X_1, .., X_n$ such that for all multisets $X_i, \sum_{x \in X_i} x = w$?

Garey and Johnson [14] show that 3-Partition is NP-complete in the strong sense, meaning that even if the natural numbers in X are bounded by a polynomial in n the problem is still NP-hard. Furthermore, we note that because of the constraints on the elements of X it follows that each X_i has to contain exactly three elements.

Theorem 14 Adjacency Preserving Grid Embedding *is NP-complete.*

Proof. Given a planar graph $G = (V, E)$, a grid graph $H = (N, Z)$, and an embedding ϕ of G into H, it can be verified that ϕ preserves at least k adjacencies from G by simply checking each edge $e \in E$. Clearly this can be done in polynomial time, which means Grid Embedding is in NP. What remains to show is that the problem is NP-hard. We show this by reduction from 3-Partition.

Given a multiset X of $3n$ natural numbers that are bounded by a polynomial in n and a target weight w, we construct a graph $G = (V, E)$ and a grid $H = (N, Z)$ such that the embedding $\phi : G \hookrightarrow H$ preserves $|E|$ adjacencies if and only if X can be partitioned into n multisets of size 3 and weight w. The graph G consists of $3n + 1$ components. For each number $x \in X$ we add a simple path of length x. The one remaining component is a large component, leaving n open holes of size w, that can only be embedded in a specific way.

We start by defining the grid graph H. We use a grid of size $R \times 3n+2$, where $R = \max(w + 4, 3n + 3)$. So let $H = (N, Z)$ be a grid graph with

$$N = \{u_{r,c} \mid 1 \leq r \leq R \wedge 1 \leq c \leq 5m\} \qquad \text{and}$$
$$Z = \{(u_{c,r}, u_{c+1,r}) \mid 1 \leq c < 3n+2 \wedge 1 \leq r \leq R\} \cup \{(u_{c,r}, u_{c,r+1}) \mid 1 \leq c \leq 3n+2 \wedge 1 \leq r < R\}.$$

Next, we define some basic components to construct the planar graph G. Let $S = (V_s, E_s)$ be a *separator* component consisting of the set of vertices

$$V_s = \{v_{c,r} \mid 1 \leq c \leq 3n+2 \wedge 1 \leq r \leq R\} \setminus \{v_{3j,r} \mid 1 \leq j \leq n \wedge 3 \leq r < 3+w\}$$

and the set of edges

$$E_s = \{(v_{c,r}, v_{c+1,r}) \mid 1 \leq c < 3n+2 \wedge 1 \leq r \leq R \wedge \{v_{c,r}, v_{c+1,r}\} \subset V_s\} \cup \{(v_{c,r}, v_{c,r+1}) \mid 1 \leq c \leq 3n+2 \wedge 1 \leq r < R \wedge \{v_{c,r}, v_{c,r+1}\} \subset V_s\}.$$

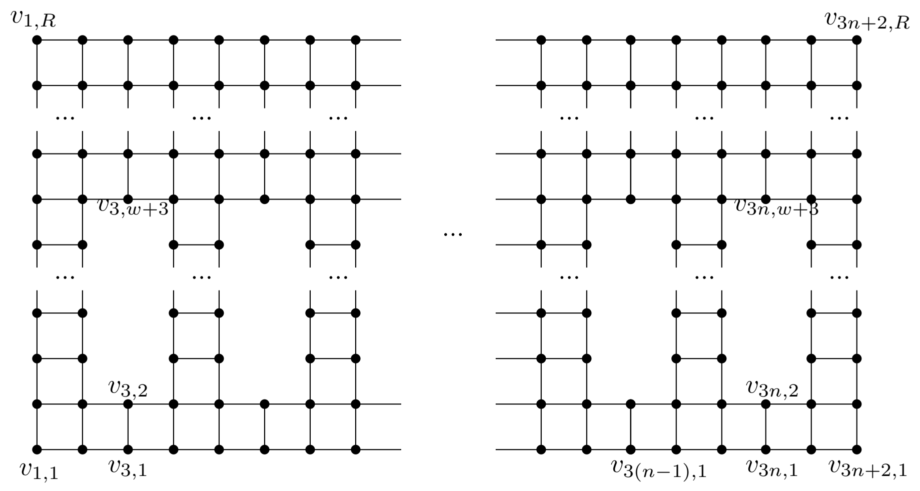


Figure 3.4: The separator component S.

Figure 3.4 depicts this component. The separator S consists of a number of $C_4 = (\{w_i \mid 0 \leq i < 4\}, \{(w_i, w_{(i+1) \bmod 4}) \mid 0 \leq i < 4\})$ graphs (i.e. a number of simple cycles of length 4). Each C_4 is connected to at least one other C_4 by at least two edges. So once we choose the position of one C_4 this fixes the position of the entire component. This means there are only four possible placements of S that preserve all adjacencies, and they only differ by rotation of $c \cdot 90°$ for $1 \leq c \leq 4$. We say the placement drawn in Figure 3.4 is the separator's *natural orientation*.

Additionally, we define a *chain* component $C(i) = (V_i, E_i)$ with vertices $V_i = \{v_{i,j} \mid 1 \leq j \leq i\}$ and edges $E_i = \{(v_{i,j}, v_{i,j+1}) \mid 1 \leq j < i\}$. Hence, $C(i)$ a simple path of length i.

Next, we define the planar graph $G = (V, E)$ consisting of the separator S and a chain $C(x)$ for each number $x \in X$. Note that we can make sure there are as many copies of $C(x)$ in G as there are copies of x in X by simply using unique vertex labels for each component. This gives us the graph G defined as:

$$G = (V, E) = \left(V_s \cup \bigcup\nolimits_{x \in X} V_x, \quad E_s \cup \bigcup\nolimits_{x \in X} E_x \right)$$

The size of G depends on the numbers in the set X. However, as all these values are bounded by a polynomial in n it follows that G is polynomial in size and we can construct it in polynomial time. A similar argument holds for H.

We now show that G has an embedding ϕ into H that preserves all $|E|$ adjacencies if and only if X has a 3-partition. For the 'only if' direction assume ϕ is an embedding of G into H that preserves all adjacencies from G. There are only four possible orientations for embedding S in the grid. Since the grid graph H has only $3n + 2$ columns the only two orientations that preserve all adjacencies in S are the natural orientation and the natural orientation rotated by 180°. It follows that the columns $3j$, for $1 \leq j \leq n$, all contain w consecutive vertices which are not occupied by S. In case S is placed in its natural orientation these are the vertices $u_{3j,3}, .., u_{3j,w+2}$, and otherwise the vertices $u_{3j,R-1-w}, .., u_{3j,R-2}$. We call this set of vertices $F(j)$. It follows ϕ places (the vertices from) all chain components on the vertices from $\bigcup_{1 \leq j \leq n} F(j)$.

Since ϕ preserves all adjacencies in G it also preserves the adjacencies in a chain $C(x)$. This means $C(x)$ in its entirety is placed on consecutive vertices of some $F(j), 1 \leq j \leq n$. Furthermore, as $|V| = |N|$ it follows that for every vertex $u \in N$ there is a vertex $v \in V$ that is mapped to u. Hence every vertex in $F(j)$ is occupied.

Now consider the partition $X_1, .., X_n$ of X that places x in set X_j if and only if the vertices from $C(j)$ are mapped onto $F(j)$. Each X_j has total weight w and contains exactly 3 elements (this follows from the bounds on elements in X). Hence $X_1, .., X_n$ is a valid 3-partition. This completes the only if direction of the proof. Figure 3.5 shows an example embedding as used in the reduction procedure.

For the if direction suppose $X_1, ..., X_n$ is a 3-partition of X. For each $X_j = \{x, y, z\}$, $1 \leq j \leq n$, we map $C(x)$ to the vertices $u_{3j,3}, ..,$ $u_{3j,x+2}$, $C(y)$ to the vertices $u_{3j,x+3}, .., u_{3j,x+y+2}$, and $C(z)$ to the vertices $u_{3j,x+y+3}, .., u_{3j,w+2}$. Finally, we place the separator S in its natural orientation such that $v_{1,1}$ is mapped to $u_{1,1}$. It is easy to see that such an embedding preserves all adjacencies. Hence G can be embedded into H if and only if X has a valid 3-partition. We conclude Adjacency Preserving Grid Embedding is NP-hard. This completes the proof. □

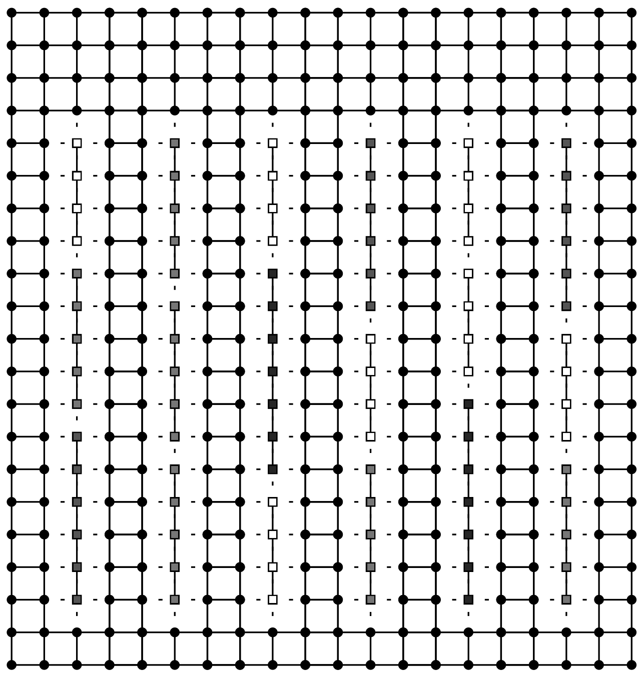
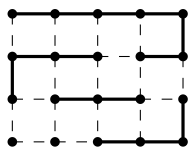


Figure 3.5: An embedding of G into H that solves the 3-Partition problem for $X = \{4^7, 5^6, 6^3, 7^2\}$ with $w = 15$. The dotted edges are edges from H that are not used in the embedding.

An exact algorithm for preserving Adjacencies. Since maps usually only have reasonably few regions we can still try to solve the Adjacency Preserving Grid Embedding problem exactly, even though it is NP-hard. We can use one of the existing algorithms to compute a maximum common subgraph of $G = (A, E)$ and $H = (B, Z)$. Conte, Guidobaldi, and Sansone [11] and Bunke et al. [7] do an experimental analysis of several maximum common subgraph algorithms on labelled graphs. For sparse graphs with few labels McGregor's algorithm [24] performs best. The McGregor algorithm is a backtracking algorithm that considers a current common subgraph and tries to extend it with all feasible pairs of edges. The resulting maximum common subgraph yields a partial one-to-one matching between A and B. The remaining vertices from A are matched to arbitrary remaining vertices from B (or instead of matching the remaining vertices arbitrarily we can also optimise one of the other criteria). The one-to-one matching that we obtain this way maximises the number of preserved adjacencies. Unfortunately, our experimental analysis (see Chapter 4) shows that this approach is infeasible in practice.

A constant factor approximation algorithm. Alternatively we can approximate the Maximum Adjacency Preserving Grid Embedding problem. However, Kann [22] shows that for a number of variants of the maximal common subgraph problem it is also NP-hard to approximate the optimal solution within a constant factor. We show that there does exist a constant factor approximation algorithm for MAXIMUM ADJACENCY PRESERVING GRID EMBEDDING. This algorithm decomposes G into a set P of vertex-disjoint edge-maximal paths. It then concatenates all paths in P into one long chain, which can easily be embedded into H while preserving all adjacencies. The global algorithm looks as follows.



```
Algorithm PATHPRESERVE($G, H$)
Input. Planar graph $G = (V, E)$ and grid graph $H = (N, Z)$ with $R$ rows and $C$ columns
Output. An embedding $\phi : G \hookrightarrow H$
1.    $U \leftarrow V$ ; $P \leftarrow \emptyset$
2.    while $U \neq \emptyset$ do
3.        Pick a vertex $v \in U$ and remove it from $U$
4.        $p \leftarrow [v]$ ; $u \leftarrow v$
5.        while $\exists w \in U$ such that $(u, w) \in E$ do
6.            Append $w$ to $p$ and remove it from $U$
7.            $u \leftarrow w$
8.        Repeat the while loop on lines 5 - 7 but instead
          prepend $w$ to $p$.
9.        Add $p$ to $P$
10.   Concatenate all paths in $P$ to one chain $C$.
11.   Let $\phi$ be the embedding that places $C[rC + 1], .., C[rC + C]$ onto row $r + 1$ for even $r$, and
      $C[rC + 1], .., C[rC + C]$ in reverse order onto row $r + 1$ for uneven $r$.
12.   return $\phi$
```

Next, we show PATHPRESERVE is a 4-approximation algorithm. The key observation in this proof is that any vertex in H has degree at most four. Hence any algorithm can preserve at most four edges incident to a vertex.

Theorem 15 PATHPRESERVE *is a 4-approximation for* MAXIMUM ADJACENCY PRESERVING GRID EMBEDDING.

Proof. Let E_P be the set of edges that is preserved in the grid embedding computed by the PATHPRESERVE algorithm and let E_O be the set of edges that is preserved in an optimal grid embedding.

We show that we can partition E into $E_1, .., E_m$ such that from each E_i the optimal solution E_O contains x_i edges and E_P contains $x_i/4$ edges. It follows that PATHPRESERVE is a 4-approximation algorithm.

We show that such a partition of E exists by constructing a partition of the vertices V into $V_1, .., V_m$ and identifying a set E_i of edges associated with each V_i. We construct these partitions as follows (see also Figure 3.6).

Let V' be the set of unpartitioned vertices, initially V, and E' the set of unpartitioned edges E', initially E. Furthermore, let $A(v) = E_P \cap \{(v,u) \mid u \in E'\}$ denote the set of unpartitioned edges incident to vertex v that are preserved in E_P. To construct new sets V_i and E_i we now pick an unpartitioned vertex $v \in V'$. In case $|A(v)| < 2$ we consider the set of vertices $V_v = V_i = \{v\} \cup (\mathit{Neigh}(v) \cap V')$ consisting of v and its unpartitioned neighbours, and a set of unpartitioned edges $E_v = E_i = \{(u,w) \mid u \in V_v \wedge w \in V\} \cap E'$ incident to vertices in V_v. For the case $|A(v)| = 2$ we simply choose $V_v = V_i = \{v\}$ and $E_v = E_i = \{(v,u) \mid u \in V'\}$. We repeat this procedure until there are no more unpartitioned vertices. Clearly this yields a partition of the set of vertices, and since all edges are incident to vertices we also obtain a partition of the edges.

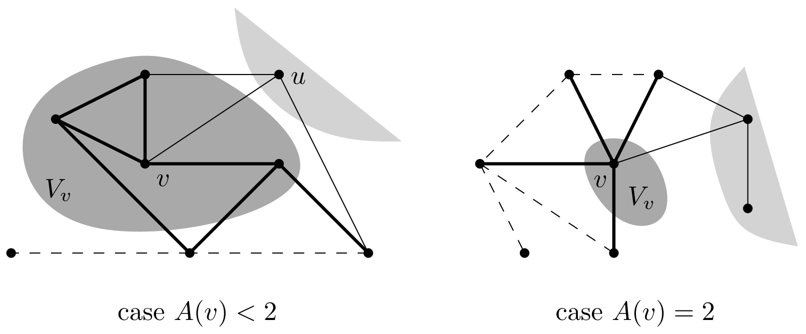
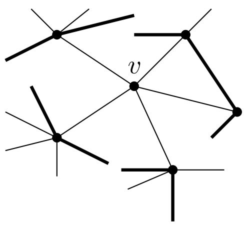


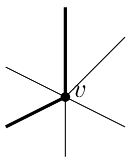
Figure 3.6: The construction of the partition. The edges E_v corresponding to V_v are drawn fat. The regular edges (for example (u,v)) are already used, dashed edges are still unpartitioned.

We now show that if the optimal solution preserves x_i edges from $E_i = E_v$ then PathPreserve preserves $x_i/4$ edges. We consider three cases based on $|A(v)|$:

Suppose $|A(v)| = 0$. This means $[v]$ is a path in P. Using that all paths in P are maximal it follows that all neighbours of v, so in particular those in V_v, have two incident edges in E_P. Since a neighbour u may be connected to another neighbour w we obtain that each neighbour preserves at least one unique edge. This means E_P contains at least $|V_v|-1$ edges from E_v. Each vertex in the grid has at most four neighbours, so the optimal solution E_O contains at most $4(|V_v|-1)$ edges from E_v.



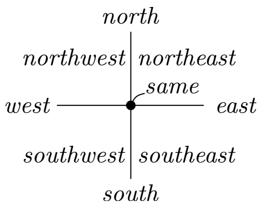
For the case $|A(v)| = 1$ we again use that all paths in P are maximal. A similar argument as in the case $|A(v)| = 0$ gives us that E_P preserves at least $|V_v|-1$ edges from E_v whereas E_O preserves at most $4(|V_v|-1)$. Finally, we have the case $|A(v)| = 2$. The optimal solution can preserve at most four edges incident to v. Since v is the only vertex in V_v it follows that E_O preserves at most 4 edges from E_v, whereas E_P preserves two.



We conclude that for each set E_i in the partition E_P preserves $x_i/4$ edges. Hence $|E_O| \leq 4|E_P|$. This shows that PathPreserve is a 4-approximation algorithm and concludes the proof. □

3.3 Preserving directional relations

Our third and last criterion is that the matching preserves the directional relation between pairs of points. We introduce the function $dir(p, q)$ to denote the directional relation of q with respect to p. We define this function as follows:



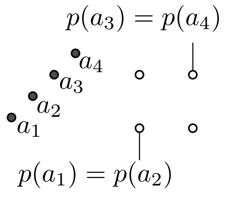
$$dir(p,q) = \begin{cases} \mathit{same} & \text{if } p = q \\ \mathit{north} & \text{if } p_x = q_x \wedge p_y < q_y \\ \mathit{northeast} & \text{if } p_x < q_x \wedge p_y < q_y \\ \mathit{east} & \text{if } p_x < q_x \wedge p_y = q_y \\ \mathit{southeast} & \text{if } p_x < q_x \wedge p_y > q_y \\ \mathit{south} & \text{if } p_x = q_x \wedge p_y > q_y \\ \mathit{southwest} & \text{if } p_x > q_x \wedge p_y > q_y \\ \mathit{west} & \text{if } p_x > q_x \wedge p_y = q_y \\ \mathit{northwest} & \text{if } p_x > q_x \wedge p_y < q_y. \end{cases}$$

The goal is now to find a matching ϕ^* that maximises the number of pairs $(a, a') \in A \times A$ for which $dir(a, a') = dir(\phi^*(a), \phi^*(a'))$. Stated differently, we are looking for the matching ϕ^* that minimises the number of out-of-order pairs W defined as

$$W(\phi) = |\{(a, a') \mid (a, a') \in A \times A \wedge dir(a, a') \neq dir(\phi(a), \phi(a'))\}|.$$

Next, we describe the global approach in our algorithm DirRel-Preserve that computes a matching that approximately minimises W. We consider the *x-order* and the *y-order* of the set of blue points A. The x-order $\pi_1, .., \pi_n$ is the order for which $\pi_i = a$ if and only if $x\text{-}rank(a) = i$, with

$$x\text{-}rank(a) = |\{a' \mid a'_x < a_x \vee (a'_x = a_x \wedge a'_y \leq a_y)\}|.$$



The x-rank denotes the number of points to the left of a (together with the points that are below a if they have the same x-coordinate). The y-order and y-rank are defined similarly. With these two orders we now define the *preferred location* $p(a)$ of a point a. The preferred location $p(a) = (r, c)$ indicates that with respect to the x- and y-order a is ideally mapped to the grid point on row r and column c. More formally, for a grid of size $R \times C$ we define $p(a)$ as follows

$$p(a) = (p(a)_r, p(a)_c) = \left(\left\lceil \frac{y\text{-}rank(a) - 1}{R} \right\rceil, \left\lceil \frac{x\text{-}rank(a) - 1}{C} \right\rceil \right).$$

Note that the preferred location is not necessarily a one-to-one matching. It may be the case that multiple points from A have the same preferred location.

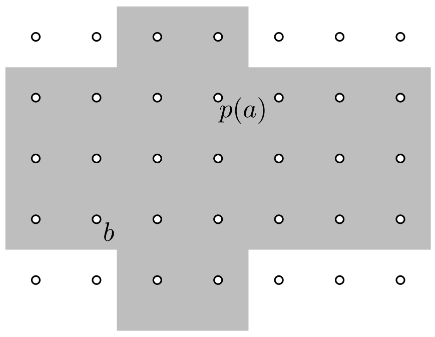


Figure 3.7: The distance $w(a,b)$ corresponds to the size of the set of grid points in the cross shape defined by $p(a)$ and b, i.e. the number of grid points in the grey area.

The notion of preferred location allows us to define a distance measure $w(a,b)$ between points $a \in A$ and $b \in B$. Let $\ell(b) = (\ell(b)_r, \ell(b)_c) = (r,c)$ denote that point $b \in B$ is the grid point on row r and column c. We define

$$w(a,b) = (|p(a)_r - \ell(b)_r| + 1) \cdot C + \\ (|p(a)_c - \ell(b)_c| + 1) \cdot (R - |p(a)_r - \ell(b)_r|)$$

as the size of the set $C(a)$ of grid points in the cross shape defined by $p(a)$ and b (see Figure 3.7). We now compute the minimal distance matching between A and B using w as distance measure. We conjecture that the resulting matching approximately minimises W:

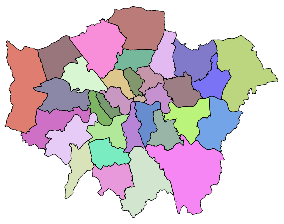
Conjecture 1 *Given a set A of n points in the plane and a set B of n grid points in an $R \times C$ grid,* DirRel-Preserve *computes a matching ϕ such that $W(\phi) \leq 4 \cdot \min_{\phi^* \in \Phi} W(\phi^*)$.*

In other words, we conjecture that DirRel-Preserve is a 4-approximation algorithm.

The intuition behind this approach is that if a point a has x-rank i, there are $i-1$ points to the left of a. Hence if we map a to a grid point b, there need to be $i-1$ grid points to the left of b. However, if there are instead $i-1 \pm \delta$ grid points to the left of b then there are at least δ points on the wrong side of b. A similar reasoning holds for the y-rank of a. With this we can show that for any point a' that is matched to a grid point $b' \in C(a)$ the directional relation is wrong. Hence the distance $w(a,b)$ is a lower bound for the number of out-of-order pairs involving a if we match a to b. What remains is to give an upper bound for the number of pairs with the wrong directional relation in terms of the size of the cross shape $C(a)$.

3.4 Choosing a grid

One topic we have not discussed until now is how to choose a suitable grid. For a map with n regions we need a grid with $m \geq n$ grid cells. However, there are infinitely many such grids. In case we restrict ourselves to grids of exactly n points there are at most $2\sqrt{n}$ different grids. However, in this restricted case most of the $2\sqrt{n}$ grids have large aspect ratios, for example $1 \times n$ or $n \times 1$, and are ill suited for our geographic grid embeddings (see Chapter 4 for examples). Furthermore, much more suitable grids might be excluded. Consider for example the map of the London boroughs. There are 33 boroughs, which means the only grids with exactly 33 cells are the 33×1, 1×33, 11×3, and 3×11 grids. However, a more square grid seems much more suitable: for example a 6×6 grid from which we remove 3 cells. Our experiments confirm this (see Chapter 4).



When using a grid of size $m > n$ we need to select a subset of n grid cells. Choosing the optimal size $m > n$ as well as an optimal subset of size n are two challenging problems. The selected subset should at least be simply connected and if possible represent the shape of the country in question. We currently cannot claim a solution for the general problem, but the following heuristic might work well in practice.

To determine the grid size we compute the rectangular dual of the map. We then use the maximum number of rectangles encountered when walking from left to right (top to bottom) as the width (height) of the grid. To select a suitable subset of the grid cells we can consider a process in which we shrink the map and remove the cells that become empty first. We note that additional research is required to evaluate the quality of these methods.

4 Evaluation

In this chapter we present an experimental evaluation of our geographic grid embeddings. We give both a quantitative and qualitative analysis of the resulting visualisations. For the quantitative analysis we compare the total distance of the matching, the percentage of pairs with the correct directional relation, and the percentage of the adjacencies that have been preserved. For the qualitative analysis we overlay the map with a colour gradient, and compare the gradient in the original map with the gradient in the resulting visualisation.

Implementation. To evaluate our geographic grid embeddings we developed a tool which can read maps in Ipe 7 XML format [8] and computes a geographic grid embedding. The resulting embedding is again stored in an Ipe 7 XML file. The tool is implemented in Scala and uses lpsolve [5] to solve the underlying linear programming problems. It currently supports the following methods to compute a geographic grid embedding:

L1_trans This method uses the approach described in Section 3.1.2 to compute an embedding that minimises the total distance under translation with the L_1 distance as underlying metric.

L1_scale This method computes the embedding that minimises D_Λ with the L_1 distance as underlying metric. It uses the approach described in Section 3.1.2.

L22 This method computes an embedding that minimises $D_\mathcal{T}$ with the L_2^2 distance as underlying metric (Section 3.1.3).

adjacency This method uses the approximation algorithm from Section 3.2 to compute a matching that approximately maximises the number of adjacencies preserved.

For the L_1 metric we can minimise the total distance under either translation ($D_{\mathcal{T}}$) or scaling (D_{Λ}). For L_2^2 we can minimise only the total distance under translation ($D_{\mathcal{T}}$). Our initial experiments showed that to obtain a good matching it is critical that the input map and the grid are roughly the same size. We therefore always start by scaling the grid such that it has the same size bounding box as the original input map. This does not guarantee to give an optimal solution but works reasonably well in practice.

We also implemented the McGregor [24] algorithm for computing the maximum common edge subgraph so we could compute the matching that preserves the maximum number of adjacencies exactly. Unfortunately even for maps with very few regions this approach was computationally infeasible. For the map with the 12 provinces of the Netherlands the algorithm already takes several hours.

Measuring the quality of geographic grid embeddings. We evaluate the quality of our geographic grid embeddings by quantitative as qualitative quality criteria. We use the following three quantitative criteria:

- the total distance of the matching using the L_1 and L_2^2 metric,
- the percentage of pairs of regions that have the correct directional relation. The directional relation between the regions is computed using the cone based model by Haar [16].
- the percentage of adjacencies from $G = (A, E)$ that are preserved by the embedding into $H = (B, Z)$: i.e. the number of edges $(a, a') \in E$ for which $(\phi(a), \phi(a')) \in Z$. Two regions are considered adjacent if and only if they share a border. We compute this by checking if the intersection of the two closed polygons representing the regions is non-empty.

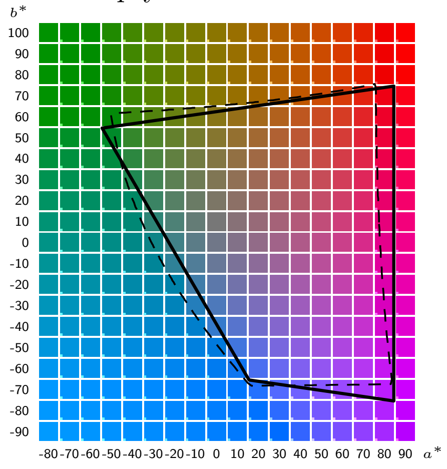


Figure 4.1: The CIE L*a*b* colour space for L* = 50. The area enclosed by the dashed lines indicates the area for which the CIE L*a*b* colours are properly defined. The area enclosed by the fat lines is the space that we overlay on the map[1].

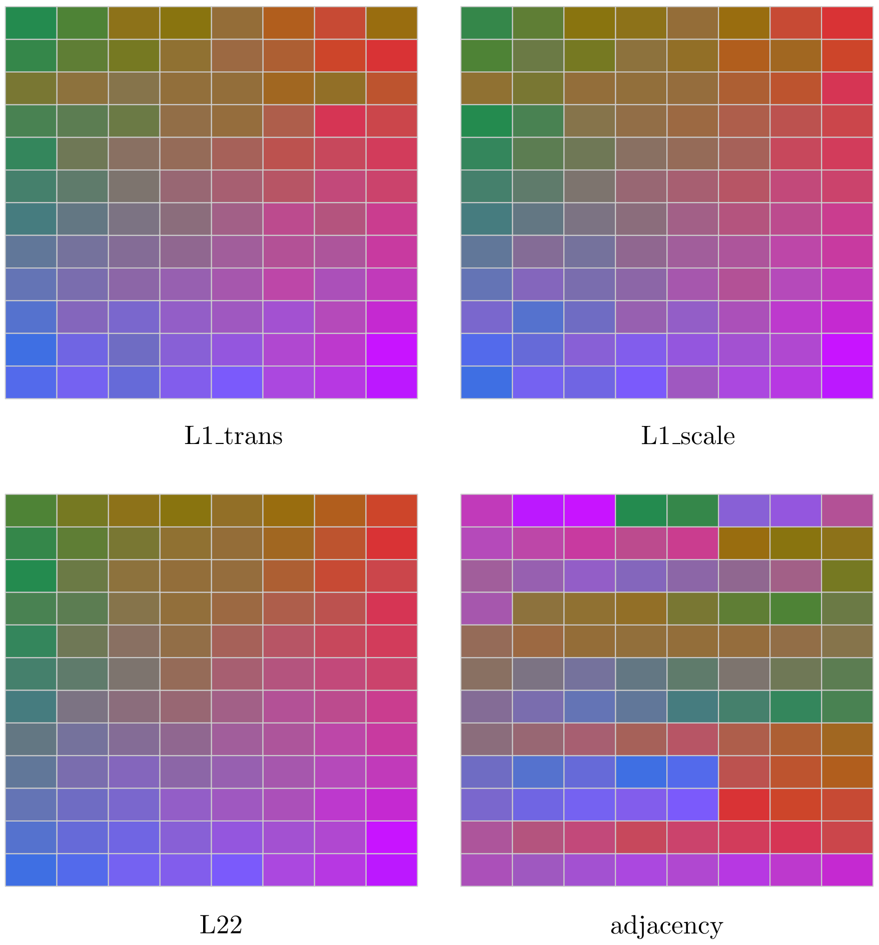
In the adjacency method we decompose the dual graph into a set of vertex-disjoint edge-maximal paths. The method still leaves some choice how to pick the next vertex on a path. Our method randomly picks a

[1]Image based on `http://www.fho-emden.de/~hoffmann/cielab03022003.pdf`

suitable vertex. We therefore run the adjacency method 20 times and use the average values in our analysis.

Additionally, we use a qualitative analysis of the resulting grid based on the colouring of the regions. Similar to Wood and Dykes [34], we map the CIE L*a*b* colour space for $L^* = 50$ onto our map (see Figure 4.1). We identify the a* axis with the x-axis and the b* axis with the y-axis and compute the colour for a given point by bilinear interpolation.

The departments of France. We use our geographic grid embedding technique on the 96 departments of France. The results of the qualitative analysis are shown in Figure 4.2. We see the grid for the L22 method shows the most natural change of colours. This also shows in the quantitative analysis (Table 4.1). The L22 method preserves the most adjacencies, it has the most pairs with the correct directional relation, and it has the smallest the L_2^2 distance.



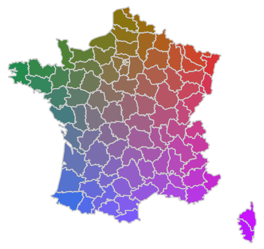


Figure 4.2: The optimal grids for the 96 departments of France for different measures.

Table 4.1: The optimal results for different measures to compute a geographic grid embedding.

Measure	Distance		Directional Rel.		Adjacencies	
	L_1	L_2^2	#	%	#	%
L1_trans	9026.64	788103.71	6560	71.93%	109	45.61%
L1_scale	9294.08	775423.66	6638	72.79%	105	43.93%
L22	9082.00	655798.18	7140	78.29%	127	53.56%
adjacency	33498.70	8533680.32	1185	12.99%	85	35.56%

Figure 4.2 shows that both the L1_trans and L1_scale method have some out-of-order cells (mainly in the upper right and upper left corner). This also shows in the quantitative results: the number of correct directional relation pairs and the number of preserved adjacencies are slightly less than for the L22 method.

In the grid embedding computed by the adjacency method the colours often change suddenly. The same results are visible in the quantitative analysis. These results are not entirely surprising since the PathPreserve algorithm does not try to preserve more than two adjacencies per cell. Additionally, the algorithm does not take the location of the regions into account. The quantitative analysis also indicates that a 4-approximation algorithm for preserving adjacencies is not sufficient since we observe in our experiments that the methods that focus on minimising the distance already preserve almost half the adjacencies.

Table 4.2: The results for different grid sizes using the L22 measure.

Grid size	Distance		Directional Rel.		Adjacencies	
	L_1	L_2^2	#	%	#	%
12×8	9234.70	622337.90	7140	78.29%	128	53.56%
8×12	9374.84	632905.18	6972	76.45%	117	48.95%
16×6	9367.98	635007.33	6966	76.38%	110	46.03%
6×16	9370.41	635873.83	7048	77.28%	101	42.26%
24×4	9513.38	664012.44	6690	73.36%	88	36.82%
4×24	9726.77	672187.95	6640	72.81%	70	29.29%
32×3	9567.07	702114.29	6410	70.29%	70	29.29%
3×32	10147.81	727028.42	6266	68.71%	63	26.36%
48×2	10467.50	866844.47	5418	59.41%	47	19.67%
2×48	10673.99	851302.40	5524	60.57%	44	18.41%
96×1	11978.28	1273845.27	2566	28.14%	22	9.21%
1×96	15879.70	1850449.51	1998	21.91%	29	12.13%

Next, we investigate the influence of different grid sizes on the resulting embedding. The results for the L22 method are shown in Figure 4.3. The most natural gradient occurs with the 12×8 grid. Since this is also the optimal grid in terms of distance this is not entirely surprising. The 8×12, 16×6, and 6×16 grids also show a smooth transitions in colour.

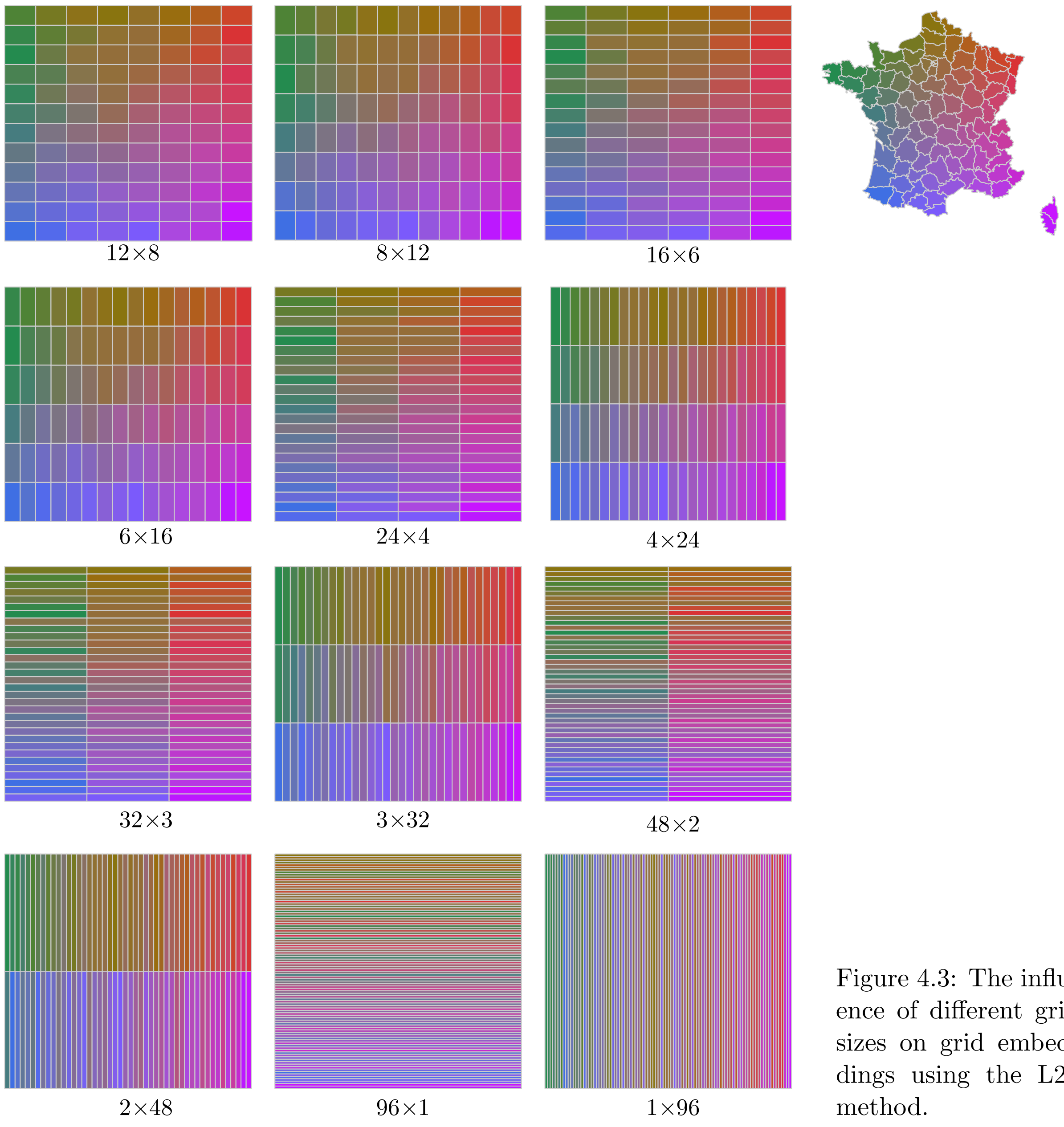
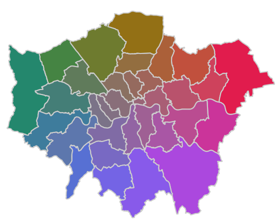


Figure 4.3: The influence of different grid sizes on grid embeddings using the L22 method.

Starting at the 24×4 grid we see more sudden colour changes, which indicates the quality of the matching degrades. This is also visible in the results of the quantitative analysis shown in Table 4.2. With these grids there is the additional problem of the aspect ratio of the grid cells. As the aspect ratio grows the cells become less square, and therefore most likely less suited to display data. We see similar results for the L1_trans and L1_scale methods. For the adjacency method the grid size does not influence the results. This was to be expected.

London Boroughs. We continue the analysis of the influence of different grids and grid sizes by looking at a map of the 33 London boroughs. Figure 4.4 shows the results for various grid sizes using the L22 method. The 3×11 grid is the best grid found by our algorithm. The other grids are all instances of a 6×6 grid in which we manually removed three grid cells. The 6×6 grids all have better results (on all three criteria) than the 3×11 grid (see Table 4.3). We can also see the set of excluded cells significantly influences the total distance of the matching. What is interesting to see is that the grids that minimise the distance not necessarily have the best results for the other criteria. Therefore we cannot really pinpoint the "best" embedding.



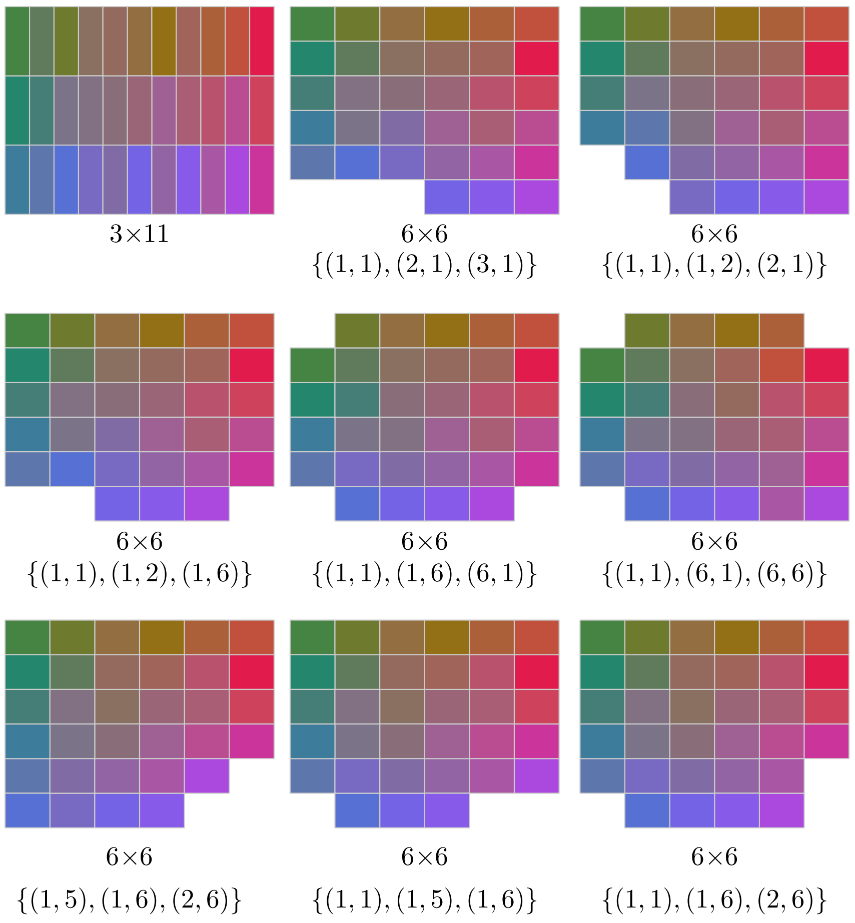
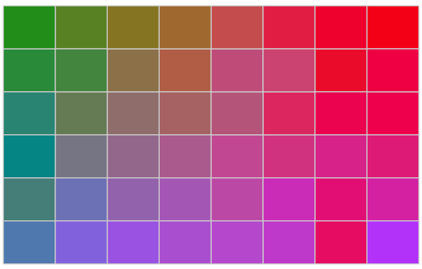


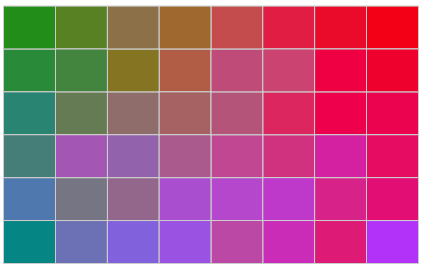
Figure 4.4: Various grids for the London Boroughs with the L22 method. For the 6×6 grids we also denote the set of removed cells.

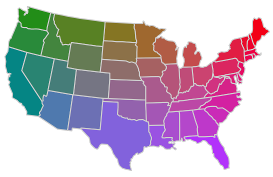
Excluded Cells	Distance		Directional Rel.		Adjacencies	
	L_1	L_2^2	#	%	#	%
3×11	3533.15	261079.75	724	68.56%	36	44.44%
$\{(1,1),(1,2),(1,3)\}$	3294.97	243011.82	752	71.21%	43	53.09%
$\{(1,1),(1,2),(2,1)\}$	3278.22	242291.38	770	72.92%	43	53.09%
$\{(1,1),(1,2),(1,6)\}$	3023.09	201956.91	770	72.92%	45	55.56%
$\{(1,1),(1,6),(6,1)\}$	3015.52	200392.36	762	72.16%	44	54.32%
$\{(1,1),(6,1),(6,6)\}$	3120.75	221855.40	730	69.13%	42	51.85%
$\{(1,5),(1,6),(2,6)\}$	2986.31	186797.48	782	74.05%	41	50.62%
$\{(1,1),(1,5),(1,6)\}$	2984.29	190687.77	758	71.78%	40	49.38%
$\{(1,1),(1,6),(2,6)\}$	2936.34	177927.18	776	73.48%	40	49.38%

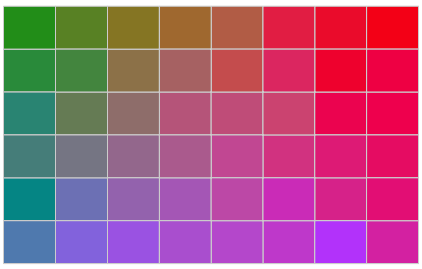
Table 4.3: The results for a 6×6 grid with different cells removed. The first line is the 3×11 grid found using the algorithm in Section 3.4.

United States. We also use our geographic grid embeddings to make a grid for the United States. In order not to artificially inflate the bounding box we only consider the 48 contiguous states. Figure 4.5 shows the resulting grid for each of the embedding methods. The results are similar to those of France: the gradient in the grid corresponding to the L22 method looks the most natural. Both the L1_trans and L1_scale methods show some out-of-order cells in the bottom right. This also shows in the quantitative results: the L22 method preserves roughly 52% of the adjacencies and 71% of the directional relations versus 49% and 68% for the L1_trans method and 44% and 62% for the L1_scale method. The adjacency method shows the same poor results as before.


L1_trans


L1_scale




L22

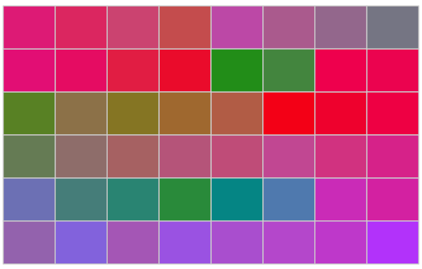

adjacency

Figure 4.5: The optimal grids for the 48 contiguous states of the U.S.

Examples. We also show some examples in which we use our geographic grid embeddings to visualise various types of data. Figure 4.6 shows our technique in combination with bar charts to display the results of the 2010 elections in the Netherlands.

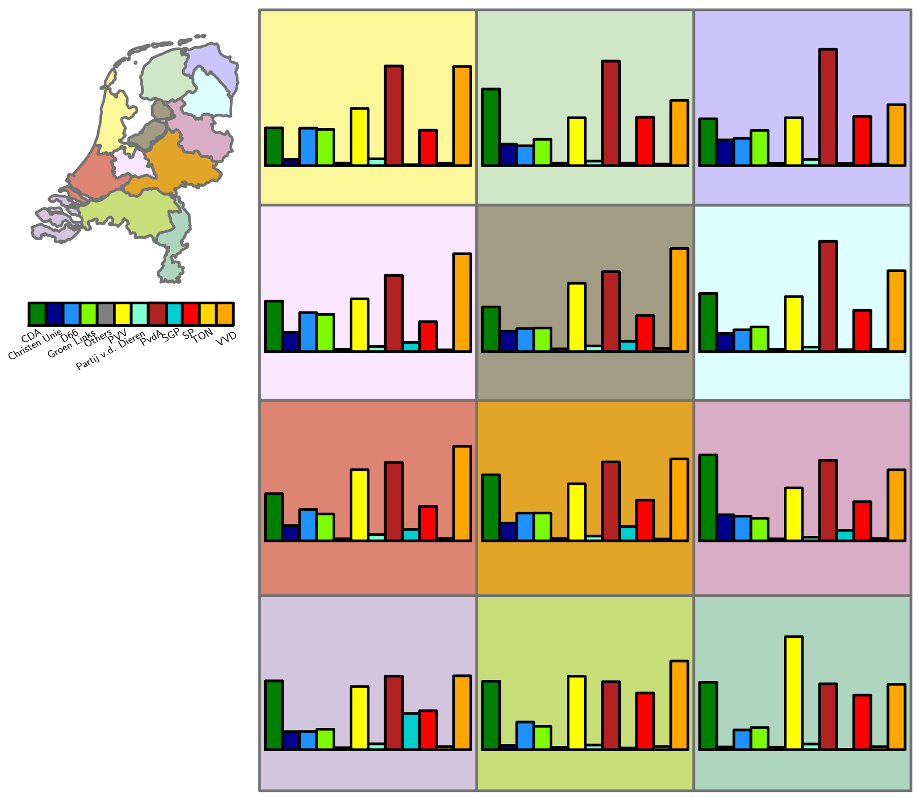


Figure 4.6: The 2010 election results in the Netherlands. Data courtesy of Election-Resources.org

Finally, in Figure 4.7 we use our technique together with a tree map to show the 2009 population estimate in the U.S. per race.

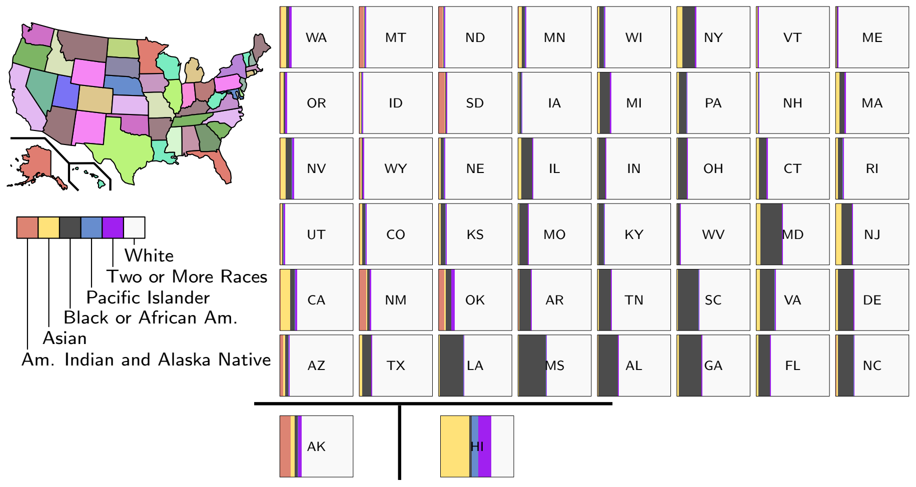


Figure 4.7: The distribution of the population of the U.S. in 2009 based on race. Data courtesy of the U.S. Census Bureau.

5 Concluding Remarks

We present a novel technique for visualising geo-referenced data. Our geographic grid embeddings technique computes an embedding of a map into a regular grid that retains key spatial properties of the regions in the map. We present three criteria for the matching between the regions and the grid cells: distance, directional relation, and adjacencies. We show how to compute optimal matchings for the distance criteria when using the L_1 Manhattan distance and the L_2^2 distance. For both distances we can compute the minimal distance matching under translation. For the L_1 distance we can also compute the minimal distance matching under scaling. Furthermore we show that it is NP-complete to determine whether there exists an embedding of a planar graph into a grid graph that preserves at least k edges. Hence we cannot hope for a fast algorithm to compute a matching that optimises the adjacencies criteria exactly. Instead we present a 4-approximation algorithm, which is mainly useful from a theoretical point of view. We also present an experimental evaluation of our approach.

There are still three important pieces missing in our work: how to compute an embedding that provably optimises for the directional relation criterion, how to get an optimal matching with respect to all three criteria, and how to determine the optimal grid and grid cells to use.

At this point we do not have a proof for Conjecture 1. This means we cannot guarantee that our method for directional relations really (approximately) maximises the number of pairs with the correct directional relation. This makes for an excellent opportunity for future work.

We know how to compute a matching that optimises one of the criteria. However, we would like a matching that optimises, or approximately optimises, all three criteria. It is not clear how we can achieve this. More research is also required to compute a matching that simultaneously minimises the distance under translation and scaling. We can also investigate if the probabilistic matching approach from Alt, Scharf, and Schymura [2] can be extended for our setting. Finally, there is the question of how to find a suitable set of grid points to use. Preferably we want an algorithm to compute an optimal solution, but alternatively we are also interested in a good heuristic.

There is also more work remaining on the practical side. It would be nice to do a more extensive experimental evaluation of our geographic grid embeddings. An interesting question is for example how our method compares to existing methods like the spatially ordered tree maps from Wood and Dykes [34], or the diagram placement algorithms from van Kreveld, Schramm, and Wolff [30].

References

[1] H. Alt and L.J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In. *Handbook of Computational Geometry*. Elsevier, 1996. Chap. 3, pages 121–153.

[2] H. Alt, L. Scharf, and D. Schymura. Probabilistic matching of planar regions. *Computational Geometry*, 43(2):99–114, 2010.

[3] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete and Computational Geometry*, 3(1):237–256, 1988.

[4] MD Atkinson. An optimal algorithm for geometrical congruence. *Journal of Algorithms*, 8(2):159–172, 1987.

[5] M. Berkelaar, K. Eikland, P. Notebaert, et al. lpsolve: Open source (mixed-integer) linear programming system. Eindhoven University of Technology. 2010. URL: `http://lpsolve.sourceforge.net/` (visited on 06/29/2011).

[6] D. P. Bertsekas. The auction algorithm: A distributed relaxation method for the assignment problem. *Annals of Operations Research*, 14(1):105–123, 1988.

[7] H. Bunke, P. Foggia, C. Guidobaldi, C. Sansone, and M. Vento. A Comparison of Algorithms for Maximum Common Subgraph on Randomly Connected Graphs. In *Structural, Syntactic, and Statistical Pattern Recognition*. Ed. by T. Caelli, A. Amin, R. Duin, D. de Ridder, and M. Kamel. Vol. 2396. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2002, pages 85–106.

[8] O. Cheong. The Ipe extensible drawing editor. 2010. URL: `http://ipe7.sourceforge.net/` (visited on 06/29/2011).

[9] S. Cohen. Finding color and shape patterns in images. PhD thesis. Stanford University, Department of Computer Science, 1999.

[10] S. Cohen and L. Guibas. The Earth Mover's Distance under transformation sets. In *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1076 – 1083, 1999.

[11] D. Conte, C. Guidobaldi, and C. Sansone. A Comparison of Three Maximum Common Subgraph Algorithms on a Large Database of Labeled Graphs. In *Graph Based Representations in Pattern Recognition.* Lecture Notes in Computer Science.

[12] J. Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards*, 69(1-2):125–130, 1965.

[13] A. Efrat and A. Itai. Improvements on bottleneck matching and related problems using geometry. In *Proceedings of the twelfth annual ACM symposium on Computational Geometry*, pages 301–310, 1996.

[14] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness.* WH Freeman & Co. New York, NY, USA, 1979.

[15] M.T. Gastner and M.E.J. Newman. Diffusion-based method for producing density-equalizing maps. *Proceedings of the National Academy of Sciences of the United States of America*, 101(20):7499, 2004.

[16] R. Haar. Computational models of spatial relations. TR-478 MSC-72-03610. Department of Computer Science, University of Maryland, 1976.

[17] M. Hagedoorn and R.C. Veltkamp. Reliable and efficient pattern matching using an affine invariant metric. *International Journal of Computer Vision*, 31(2):203–225, 1999.

[18] F.S. Hillier and G.J. Lieberman. *Introduction to mathematical programming.* McGraw-Hill, 1990.

[19] Jiawei Hong and Xiaonan Tan. A new approach to point pattern matching. In *9th International Conference on Pattern Recognition*, volume 1, pages 82 –84, 1988.

[20] D.P. Huttenlocher, K. Kedem, and M. Sharir. The upper envelope of Voronoi surfaces and its applications. *Discrete and Computational Geometry*, 9(1):267–291, 1993.

[21] D.P Huttenlocher, G.A. Klanderman, and W.A. Rucklidge. Comparing Images Using the Hausdorff Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993.

[22] V. Kann. On the approximability of the maximum common subgraph problem. In *STACS 92*. Vol. 577. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1992, pages 375–388.

[23] H.W. Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[24] J. J. McGregor. Backtrack search algorithms and the maximal common subgraph problem. *Software: Practice and Experience*, 12(1):23–34, 1982.

[25] E. Raisz. The rectangular statistical cartogram. *Geographical Review*, 24(2):292–296, 1934.

[26] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, 1992.

[27] T.A. Slocum, R.B. McMaster, F.C. Kessler, and H.H. Howard. *Thematic cartography and geovisualization.* Second Edition. Pearson Prentice Hall, 2009.

[28] J. Sprinzak and M. Werman. Affine point matching. *Pattern Recognition Letters*, 15(4):337–339, 1994.

[29] P. Vaidya. Geometry helps in matching. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 422–425, 1988.

[30] M. van Kreveld, É. Schramm, and A. Wolff. Algorithms for the placement of diagrams on maps. In *Proceedings of the 12th ACM International Symposium on Advances in Geographic Information Systems*, pages 222–231, 2004.

[31] J.J. Van Wijk and H. Van de Wetering. Cushion treemaps: Visualization of hierarchical information. In *Proceedings IEEE Symposium on Information Visualization*, pages 73–78. IEEE, 1999.

[32] R.C. Veltkamp and M. Hagedoorn. State of the Art in Shape Matching. In. *Principles of visual information retrieval.* Springer Verlag, 2001. Chap. 4, pages 87 –115.

[33] J. Wood and J. Dykes. BikeGrid: Cycle hire docking station viewer. 2010. URL: `http://www.gicentre.org/bikegrid` (visited on 06/27/2011).

[34] J. Wood and J. Dykes. Spatially ordered treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1348–1355, 2008.