FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Generating Stable Demers and Iso-Hexagon Cartograms

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Logic and Computation / Computational Intelligence

by

Soeren Nickel
Registration Number 01528974

to the Faculty of Informatics
at the TU Wien
Advisor: Assoc.Prof. Dipl.-Inform. Dr.rer.nat. Martin Nöllenburg

Vienna, 30th July, 2019

Soeren Nickel   Martin Nöllenburg

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

Erklärung zur Verfassung der Arbeit

Soeren Nickel

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 30. Juli 2019

Soeren Nickel

Acknowledgements

I'd like to thank my advisor Martin Nöllenburg who had helpful advise and guidance at every turn, Wouter Meulemans and Max Sondag who were of invaluable assistance in the implementation phase and everyone at the Algorithms and Complexity group at the TU Vienna for the great atmosphere I experienced.

Just as much I have to thank my parents, my siblings, my friends & family for putting up with me during my times of disappearing into work.

Most of all I want to thank my girlfriend Meli for all her love and support. Σ'αγαπώ.

Kurzfassung

In dieser Arbeit behandeln wir das Thema der automatischen Generierung von stabilen Demers Kartogrammen und Iso-Hexagon Kartogrammen. Ein Demers Kartogram/Iso-Hexagon Kartogram ist eine Repräsentation von Daten auf einer Karte, in welcher alle Länder als Quadrate/reguläre Sechsecke dargestellt werden und die assoziierten Daten eines Landes die Größe diese Quadrats/regulären Sechsecks bestimmen. Diese Art von Kartogrammen ist beliebt um Unterschiede zwischen der Größe eines Landes und anderen georeferenzierten Daten aufzuzeigen, sowie um die Veränderung in diesen Daten über einen gewissen Zeitraum hinweg zu visualisieren. In diesen Serien von Kartogrammen, aber auch in einem Kartogram generell, ist es von größter Wichtigkeit das mentale Modell des Betrachters so wenig wie möglich zu stören. Um dies zu erreichen, versuchen wir *stabile* Demers Kartogramme und Iso-Hexagon Kartogramme zu erstellen, die gewissen Quaitätskriterien—wie zum Beispiel benachbarte Regionen, wenn möglich, benachbart zu halten—entsprechen. Wir präsentieren darüber hinaus NP-Hardness Beweise für einige generalisierte Versionen der auftretenden Probleme sowie eine Methode, eine verlorene Adjazenz zwischen benachbarten Regionen in einem Iso-Hexagon Kartogram zu visualisieren. Und schließlich präsentieren wir eine experimentelle Auswertung unseres Models.

Abstract

In this thesis we approach the topic of automated generation of stable Demers cartograms and iso-hexagon cartograms. A Demers cartogram/iso-hexagon cartogram is a representation of a data set on a map, which represents each country as a square/regular hexagon and uses the associated data for every country to determine the size of this square/regular hexagon. These kinds of cartograms are widely used to visualize discrepancies between the size of a country and some other georeferenced data as well as to visualize the change of a data set over time. In these series of cartograms, but also in a cartogram in general, it is of vital importance to disturb the mental model of the user as little as possible. In order to achieve this, we try to create *stable* sets of cartograms, which fulfill certain quality criteria, like keeping as many adjacencies as possible. For this, we present an LP formulation which can be used to create stable sets of Demers cartograms and iso-hexagon cartograms. We further present a NP-hardness proof for a generalized version of some of the problems and present a method of visualizing lost adjacencies in a iso-hexagon cartogram. Finally we present an experimental evaluation of our model.

Contents

CHAPTER 1

Introduction

When you look out of the window you can see a great deal in an instant. The mind has an extremely powerful system for processing imagery, which can instantly analyze a pattern of colors, of light and shade and know (or at least think) that these are trees, houses or people out there. How long would it take to describe all that you can see in words?
(Dimitris Ballas and Danny Dorling, 2011, [BD11])

In 1934, Raisz [Rai34] published "The rectangular statistical cartogram". His goal was to create economic maps, that showed the distribution of manufacturing sites like steel factories, textile mills, power plants, etc., in the United States. His first approach was to take a geographically accurate map, in which the value of every state was visualized, exactly where the states lie on the map. This resulted in maps, that were too crowded to be useful in the north-east of the US, where small states with high production rates are densely packed, while other parts of the map, like the mountain states (Colorado, Wyoming, Utah, New Mexico, Nevada, Idaho, Arizona and Montana), tend to be relatively empty. The underlying problem is, that the size of a state (or country) does not necessarily correlate with its economic production, population, GDP per capita and other data. Raisz envisioned a map in which the area of a state is not defined by its actual real world shape, but every state is represented by a rectangle with an area proportional to its data value. Erwin Raisz's solution was a *cartogram*.

Cartograms are now widely used. Results from United States elections are commonly displayed in the form of a cartogram. The compact form and nice visual design make them just as appealing to online and print media, as their possibility to combine political, socioeconomic or in fact any data and their underlying geographical structure. Nusrat et al. [NAK18] give an example, shown in Figure 1.1, which illustrates how big the discrepancy between displaying information on a colored geographically accurate map—which is called a *choropleth map*—and a cartogram can be. The example shows the

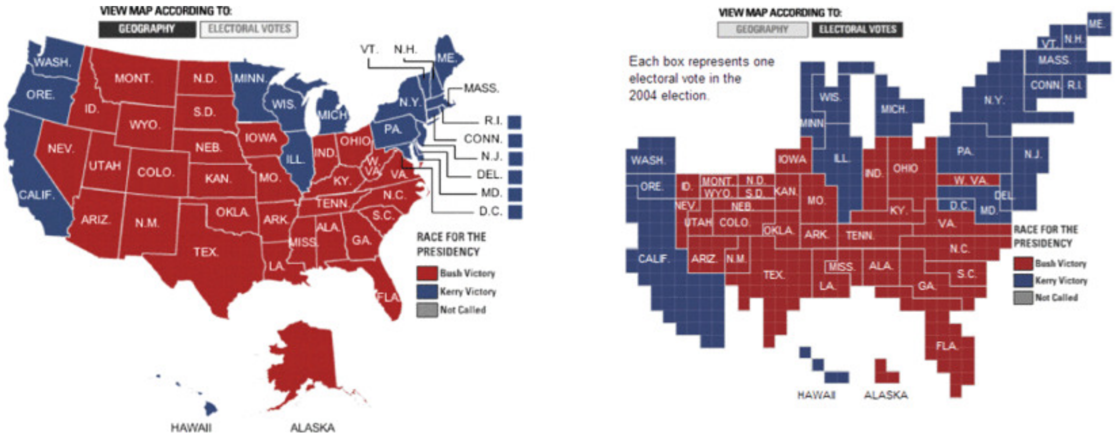


Figure 1.1: Geographic map and a cartogram for the 2004 US election [NAK18].

electoral vote distribution in the 2004 US presidential election. While the choropleth map on the left suggest an overwhelming majority in votes for George W. Bush, the cartogram on the right reveals the rather close result in relation to the electoral votes. Cartograms are regularly used to illustrate skewed distributions (in relation to geography). “World Mapper” created a cartogram in 2001 [Wor01] (Figure 1.2), which was heavily featured in online media and stage talks, that displays the scientific contributions of countries measured by the number of published scientific articles. The TED talk of Alisa Miller [Mil08] used cartograms to illustrate sharp contrast between real world events and the frequency of newspaper articles on these topics, and in the progress argued that this can distort the view of the world. A recent entry from 2018 on World Mapper depicts the occurrences of Ebola in different countries [Wor18].

Erwin Raisz’s cartogram is often a starting point when recalling the history of the cartogram [Tob04], even though he is neither the first to use the term cartogram, nor is he the first to create a map in which the area of a country is proportional to a desired value. The earliest mention of a cartogram, as recalled by Friins [Fri74], is made by Charles Joseph Minard in 1850. Several others used it to describe a statistical map [PW+32] and choropleth or statistical maps [KDW86]. An early example of a map, in which such proportional distortions are used, is “Grundys map” printed in the Washington Post in November of 1929 [wPo29]. This “map of the United States showing the size of each State on the basis of population and Federal Taxes” [wPo29] mostly keeps the shape of the states, all adjacencies can be shown, but their relative size is radically changed.

Ballas and Dorling state that, “It is increasingly and convincingly argued that conventional maps should not be used to map human data and that cartograms [...] should be used instead” [BD11]. However they also mention that among the desirable properties of a cartogram is the preservation of orientation and contiguity.

With this in mind we turn to an alternative way of representing countries, very similar to the idea of Raisz [Rai34]. Regular polygons of desired size can be used to represent each

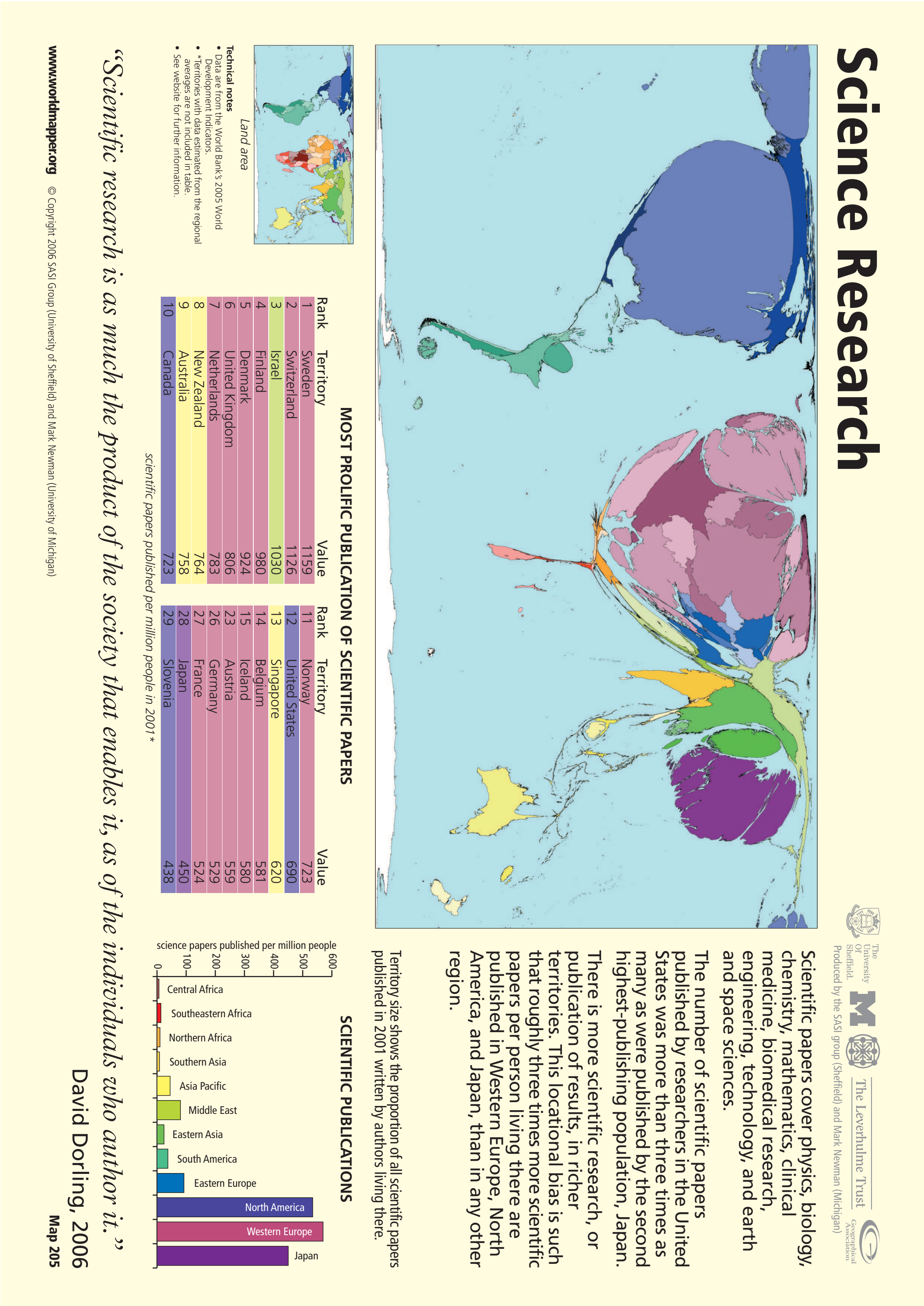


Figure 1.2: Scientific contributions of countries measured by the number of published articles of authors living in the country [Wor01].

Figure 1.3: Levasseur's map as reported by Funkhouser [Fun37].

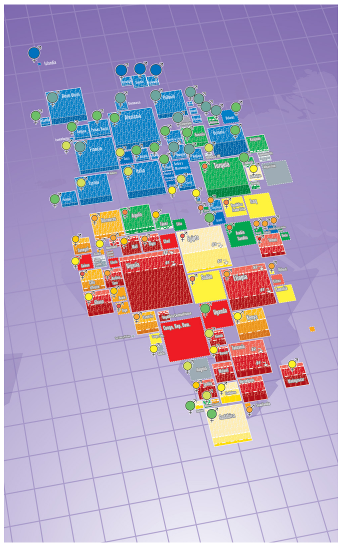


Figure 1.4: The social watch map "The earth should be blue" [Wat09].

country. Danny Dorling [Dor96] created his Dorling cartograms in an effort to reduce border complexity, by using circles. "If, for instance, it is desirable that areas on a map have boundaries which are as simple as possible, why not draw the areas as simple shapes in the first place?" [Dor96]. And while a circle might be the simplest shape possible it is not the only simple shape that can be used in such a cartogram.

An example of this is the oldest cartogram, we could find. It was created by Pierre Émile Levasseur in 1870, as reported by Funkhouser [Fun37], even though Funkhouser did not call this map a cartogram and in fact used the term cartogram to describe, what would be called a cloropleth map today. In the cartogram, depicted in Figure 1.3, every country is represented by a square, and the area of the square is proportional to the desired data value. This is, in fact, an example, that invokes the visual design of a Demers cartogram around 130 years before they were called Demers cartograms.

Similar to the Dorling cartograms [Dor96], the Demers cartograms use a fixed 2-dimensional shape of variable size in order to represent a country in the cartogram. However instead of using circles, it uses squares. The Demers cartograms were introduced by Bortins and Demers [BDC02] in 2002 on their website "Cartogram Central". The map scales each country according to its population. Bortin and Demers argue, that, due to the ability to be tighter packed than circles, the squares leave fewer gaps between the regions. This makes for a more contiguous region [BD11]. The other mentioned desired property of preserving orientation can also be met in the context of a Demers cartogram and we will discuss this in a later section in this thesis. A third property, which is beneficial to the essential use of a cartogram is comparability of regions. In this

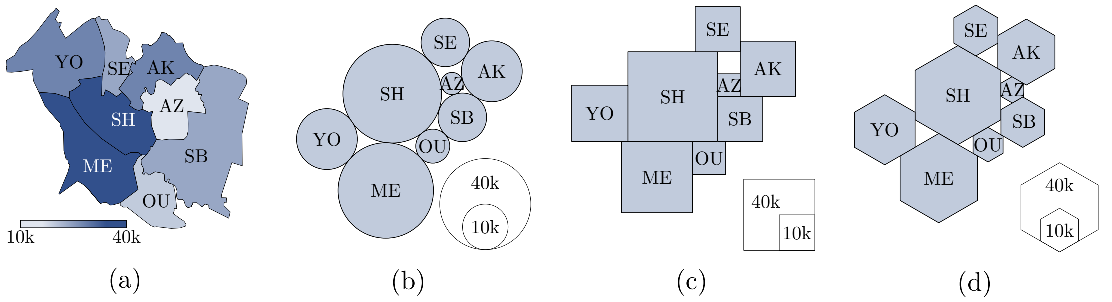
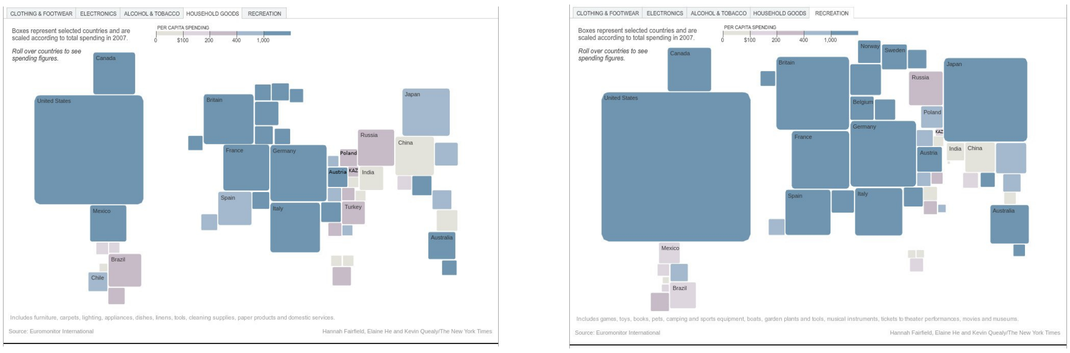


Figure 1.5: Examples of map-based visualizations. (a) Choropleth map, (b) Dorling Cartogram, (c) Demers cartogram, (d) iso-hexagon cartogram. The visualizations are only exemplary.

area, the Demers cartograms are particularly outstanding, as anyone can differentiate at first glance between squares of different size while, for example, rectangles can be misleading since their perceived size can be heavily dependent on their aspect ratio. An example of the usage of a Demers cartogram is produced in the form of an interactive map on the website “Social Watch” [Wat09], of which an example screen can be seen in Figure 1.4.

The Demers cartograms can be viewed as an adaption of the Dorling cartogram to a different, but regular shape. The extension of the Demers cartogram to yet another regular polygon comes natural. However instead of simply doubling the number of edges, we opt for the regular hexagon. The reason for this lies again in the first mentioned benefit of the Demers cartogram. The hexagon is the regular polygon with a maximal number of corners, s.t., it still allows for a gap-free tiling of the two-dimensional plane. The octagon makes this feat impossible. We therefore will look at cartograms in which every region is represented by a hexagon of appropriate size. We will call these cartograms iso-hexagon cartograms. A comparison of a standard clpopulation map, a Dorling cartogram, a Demers cartogram and an iso-hexagon cartogram, all of which visualize a fictional data set on Shibuya and its surrounding neighborhoods in Tokyo, can be found in Figure 1.5.

One common use, which we have already mentioned, is the use of cartograms to visualize the continued development of information pertaining to a certain region of the world or the world in total. Danny Dorling [Dor12] presents in an article in *The Guardian* a map, that visualizes the “space and time trend of unemployment in Great Britain, 1978-1990”, which uses the outer rings of the circles in a Dorling cartogram and shades them in different colors to visualize change over the years. Cartograms are also commonly used in interactive applications to visualize these kinds of data. The *New York Times* [Tim16] created an interactive application that scales every country, i.e., the circle of a country as this is a Dorling cartogram, proportional to the number of medals the won in a particular instance of the Olympic Games. The user can browse through the years and see the changes between the games. This was even done for Demers cartograms and again by the New York Times [FHQ08]. In this application, rather than displaying the same kind



(a) (b)

Figure 1.6: Interactive application of the New York Times, depicting purchases of (a) household goods and (b) recreational goods.

of data through time, they chose to put purchases of different goods made by countries into perspective. In this application we want to point to the importance of relative placement of regions. In Figure 1.6a we can see an example screen of the application that shows purchases of household goods, in which the squares representing Austria and Kazakhstan are placed as neighbors, with Austria to the left of Kazakhstan. When the user switches to the screen that depicts purchases made in recreation (Figure 1.6b), this relation drastically changes, with Kazakhstan now being adjacent to Japan and above of Austria. Poland, which was previously placed below the square of Japan and to the left of Russia, is now to the left of Japan and below Russia. This can potentially disturb the mental image the user has of the world and possibly interfere with the comparability of the two cartograms.

This thesis aims to present a method, which produces cartogram progressions, which preserve the mental image of the user. In order to achieve this we approach this problem first on a theoretical level in which we aim to relate the underlying algorithmic problems to already existing problems in the literature, in particular how Demers cartograms and iso-hexagon cartograms are related to the representation of a graph with touching squares and regular hexagons, known as *contact representations*. We also present a linear program model for the creation of Demers cartograms and iso-hexagon cartograms, as well as a working implementation. In the next section we will present related work in the area of cartograms and contact representations of graphs.

In Chapter 2 we present concepts, naming conventions and notation, which we will use throughout the thesis. Highly specialized concepts, which are commonly only relevant to a single chapter are sometimes presented in the chapters themselves.

In Chapter 3 we identify the underlying algorithmic problems of the creation of Demers cartograms and iso-hexagon cartograms. We also prove NP-hardness for a generalized version of the identified problems.

Next we present the theoretical model and optimization method, which is used to create the desired Demers cartograms and iso-hexagon cartograms, in Chapter 4.

At points throughout this thesis, we will encounter problems of topography, namely that countries, which are adjacent in reality, are not next to each other in a cartogram. A method of handling this problem is to connect these regions with a curve. The question whether and how this is possible is tackled in Chapter 5.

And finally we will describe our implementation and its usage in Chapter 6. We will also present an experimental evaluation of the implemented application and discuss some experimental extensions.

1.1 Related Work

In this section we present relevant related work in the areas of cartograms and contact representation of graphs.

1.1.1 Cartograms

Next to the already mentioned formal definition of Raisz [Rai34], there are newer ones by Heilmann et al. [HKPS04] and van Kreveld and Speckmann [vKS07]. Van Kreveld and Speckmann also identify four different types of cartograms. A contiguous cartogram deforms countries and enlarges or shrinks them to the desired size. Adjacencies can be maintained, while accuracy in the shape of each country is sacrificed. Most automated procedures to create cartograms focus on contiguous cartograms. Tobler [Tob73] presents an early method to create "Rubber map cartograms". Other examples of algorithms for contiguous cartograms are Dougenik et al. [DCN85], Torguson [Tor90], Edelsbrunner and Waupotitsch [EW97], Kocmoud and House [HK98], Keim et al. [KNP04], Gastner and Newman [GN04] and Inoue and Shimizu [IS06].

The second category of non-contiguous cartograms shrinks down each country such that relative sizes between countries are correct. Shape and size of a country can be realized perfectly, while adjacencies between regions are generally lost. These cartograms are easy to compute [NK16].

The third category, which are the already mentioned Dorling cartograms, can be automatically created by a force-based algorithm [Dor96]. It uses forces, which ensure disjoint circles by repelling overlapping circles away from each other, while trying to resemble the topology of the original map by applying an attractive force that pulls the circles to their original position. These forces are applied iteratively until the cartogram does not contain any overlaps between circles. Demers cartograms were proposed by Ian Bortins and Steve Demers on their website Cartogram Central [BDC02]. Even though Demers cartograms are a variant of the Dorling cartograms, we did not find a scientific result, which describes a similar force based method for them. Examples of similar force-based implementations can however be found online [Giv16]. At this point, we do not know of any publication which takes a closer look at Demers cartograms outside of Nickel et al.[NSM+19]. There exists a sketch of a theoretical formalization as a linear program, which creates stable Demers cartograms including the guarantee of keeping orthogonal separations from the input in the output [CKM+18].

Raisz map [Rai34] falls into the fourth category of rectangular cartograms. These cartograms use recursive subdivisions of a big rectangle to represent countries. The first automated method of creating these cartograms was presented by Heilmann et al. [HKPS04]. They use a genetic algorithm to evaluate multiple generations of cartograms. Algorithms by van Kreveld and Speckman [vKS05, vKS07] create a starting cartogram, based on the topology of a map. Then they move the horizontal and vertical segments of the cartogram alternatingly in order to either reach the correct sizes or topology for all countries. Speckmann at al. [SvKF06] present an alternative method to this heuristic by using a linear program.

1.1.2 Contact Representation of Graphs

A Demers cartogram tries to represent adjacencies between regions correctly by placing the corresponding squares next to each other. This has an obvious connection to the concept of graph contact representations. A famous and central theorem of this area is Koebe's circle packing theorem [Koe36], which proves that every planar graph has a representation in which all vertices of the graph are circles and two circles touch if and only if the corresponding vertices were adjacent. This is a circle contact representation. Hlineny and Kratochvil [HK01] present a survey of contact representations using disks and balls. Contact representations can also represent countries with regular or non-regular polygons and polytopes. In a *proper contact representation* the boundaries of two polygons are required to share a segment of non-zero length. Duncan et al. [DGH+11] prove that every planar graph has a proper contact representation with (non-regular) hexagons but some planar graphs do not have such a representation with pentagons.

At this point we want to emphasize that the general question whether a graph has a contact representation using a set of shapes is different from the problems considered in this thesis, since we will assume a fixed size for every individual region while the while in the classical contact representation problem the question is only if a representation exists with arbitrary size of the shape of a vertex.

Bremner et al. [BEF+12] showed that deciding if a graph admits a proper contact representation with unit cubes in 3D is NP-hard. They achieve this by modeling a logic engine using cubes. This logic engine can encode an instance of Not All Equal 3-Sat by creating a graph that admits a proper cube contact representation if and only if there is a variable assignment which makes the instance true. Kleist and Rahman [KR14] used a similar approach to model a logic engine with squares in 2D and proved the NP-Hardness of deciding whether a given graph admits a proper square contact representation

CHAPTER 2

Preliminaries

In this chapter we want to present some main concepts of graph theory and linear programming, as well as the concept of drawing dimensions.

2.1 Graphs

All graphs considered in this thesis are undirected graphs. An undirected Graph $G = (V, E)$ consists of a (finite) set of vertices V and a (finite) set of edges E. Every edge $e \in E$ is a connection of two vertices v, u, defined as the unordered set $e = \{u, v\}$, where $u \neq v$.

The vertices v, u are called *adjacent* and are being connected by e. The edge e is called *incident* to v and u. A series of vertices $p = [v_0, v_1, \ldots, v_k]$, s.t., $\forall v_i, v_{i+1} : \{v_i, v_{i+1}\} \in E$, is called a *path* or a *chain* of length $k-1$ from v_0 in to v_k in G. If the graph exhibits the property that that all pairs (u, v) are connected through a path, we call the graph *connected*, otherwise we call it *disconnected*.

2.2 Linear Programming

Linear programming is an optimization method for *continuous optimization problems*. Linear programming will be used to model several problems in this thesis. The modeling and solution of linear programming problems is a subfield of computer science on its own and we will not explore the methods which are used to solve a linear program—detailed explanations can be found in the books of Shrijver [Sch98] and Bertsimas and Tsitsiklis [BT97]—but only explain the general approach of modeling a problem to the extend needed in this thesis. Surveys on the topic have been provided by Chandru and Rao [CR10a, CR10b].

A *linear program*, or LP for short, aims to maximize or minimize a given objective function, subject to specified constraints. The formal definition is given below.

Definition 1 (Linear Program). *Let $A \in \mathbb{R}^{m \times n}$ be an $m \times n$ matrix $b \in \mathbb{R}^m$ and m-dimensional vector and $c \in \mathbb{R}^n$ an n-dimensional vector. Then the corresponding* linear program *is given as*

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{subject to } Ax \leq b \end{aligned}$$

where $x \in \mathbb{R}^n$ is an n-dimensional vector of real variables.

A linear program is not restricted to minimizing an objective function or to constraints of the form presented above. It can instead also maximize the objective function. Constraints can also specify $Ax = b$ or $Ax \geq b$. All types of constraints can be used simultaneously as they can be transformed into constraints of the other form. However all constraints must be at all times linear inequalities or equalities.

The set of all possible value combinations is called the solution space. The linear constraints restrict the solution space to a *feasible region*. Every solution contained in the feasible region is a valid solution. Conversely an empty feasible region results in an unsolvable or infeasible LP.

The aim of the LP optimization is to maximize or minimize the objective function (which is dependent on the variables in the LP) over all valid solutions. This can be done efficiently by exploiting the shape of the feasible region.

The design of the LP model includes the specification of a set of variables and constraints which are supposed to capture the requirements and optimization goals of a problem. This model is than solved. Multiple methods for finding an optimal solution for an LP exist. Among those are the simplex algorithm [Dan51] and Karmakars polynomial-time interior point method [Kar84], which are both used in practice. The commercial optimization tool IBM ILOG CPLEX, which can be used to model and solve an LP, is regularly used in industrial settings. An LP can in general be solved in polynomial time.

2.2.1 Mixed Integer Programming

A linear program can be solved in polynomial time, which makes it very appealing as a optimization method. But some restriction are inherent to the linear program. One important feature is that all variables in a linear program must be real variables. This means we can not express the constraint that a variable must be an integer number for a solution to be valid. This is possible in a variant of the linear program called the *mixed integer program*, in which we can require a subset of variables to be element of the integer numbers.

Definition 2 (Mixed Integer Program)**.** *Let $A \in \mathbb{R}^{m \times n}$ be an $m \times n$ matrix $b \in \mathbb{R}^m$ and m-dimensional vector and $c \in \mathbb{R}^n$ an n-dimensional vector. Let $J \subseteq \{1, 2, \ldots, n\}$. Then the corresponding* mixed integer program *is given as*

$$\begin{aligned} &\textit{minimize } c^T x \\ &\textit{subject to } Ax \leq b \\ &\qquad\qquad x_j \in \mathbb{Z} \quad \forall j \in J \end{aligned}$$

where $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ is an n-dimensional vector of real variables.

The important difference between a linear program and a mixed integer program is, that finding an optimal solution for a mixed integer program is NP-hard, even though modern optimization tools like the IBM ILOG CPLEX optimization studio can solve mixed integer programs reasonably fast. We will only use an mixed integer program in Section 6.3, in which we present some experimental extensions to the program, but it should nevertheless be stated that requiring variables in linear program to have integer (or boolean) values can lead to significantly increased runtimes.

2.3 Distance Metrics

The distance of two points $p = (x_p, y_p), q = (x_q, y_q)$ in the Euclidean plane is normally quite intuitively defined as the Euclidean distance of p and q. This definition of distance is also referred to as the Euclidean norm and is defined as follows.

$$d(p, q) = \sqrt{(x_q - x_p)^2 + (y_q - y_p)^2}$$

Since this calculation has the necessity of both squaring and taking the root of something, it is not suitable for usage in a linear program, as explained in Section 2.2. There are, however several other ways to define the distance of two points. In this thesis we will make use of both the L_1 and the L_∞ metric which are defined next.

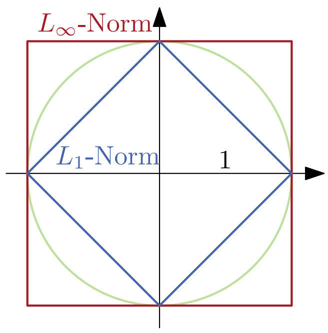


Figure 2.1: Unit circles for the Euclidean (green), the L_1- (blue) and the L_∞-Norm (red).

As the unit circle is defined as the set of all points with distance 1 from the origin of a coordinate system, we can draw unit circles for different metrics. The unit circles for the Euclidean norm as well as for both the L_1 and the L_∞ metric are displayed in Figure 2.1.

2.3.1 L_1 Metric

The L_1 metric, which also called the *Manhattan distance* and the *sum norm* is defined as the sum of the differences in both the x- and y-dimensions. Formally the distance of two points p, q in the L_1 metric is defined as follows.

$$d(p, q) = |x_q - x_p| + |y_q - y_p|$$

This metric is commonly explained by imagining the navigation on a regular street grid with square housing blocks, as they can be found in more modern cities in the US. If one would try to get from, e.g., 10^{th} and 74^{th} street to the corner of 14^{th} and 68^{th} street in Brooklyn, NY, it does generally not matter if you walk down 10^{th} avenue up to the intersection with the 68^{th} street and then down 68^{th} street up to the destination, if you first walk down 74^{th} street or if you take a combination of alternating left and right turns in between the two points. The distance is the same, i.e., 6 streets down plus 4 avenues across.

Note that the calculation of this metric does not involve squaring or the taking of a root. It does entail the calculation of an absolute value, which can be done in a linear program. If an absolute value is required to be smaller than a given threshold value, we can reformulate this into two constraints requiring bothe the positive and negative value to be smaller than the threshold. Similarly we can require both to exceed a certain threshold value to ensure an absolute value bigger than the threshold. It is therefore possible to calculate an L_1 distance in a linear program.

2.3.2 L_∞ Metric

In the Euclidean norm, changes in the coordinate with the bigger difference between the two points p, q has a bigger impact on the distance than changes in the coordinate with the smaller difference. The L_∞-metric in takes this to the extreme in a sense. The distance between two points is exclusively defined by the difference in the dimension which has the biggest difference. Formally this means:

$$d(p, q) = \max(|x_q - x_p|, |y_q - y_p|)$$

It is important to note that in using the L_∞-metric, two touching squares a, b have a distance of exactly $\frac{w_a + w_b}{2}$, where w_a and w_b are the edge lengths of the two squares. The placement of the two squares is irrelevant. Note that this does not happen for the L_1-metric.

2.4 Drawing Dimensions

At multiple times throughout this thesis we will refer to the drawing dimensions of a cartogram. This refers to either the commonly used x- and y-dimensions in a Cartesian

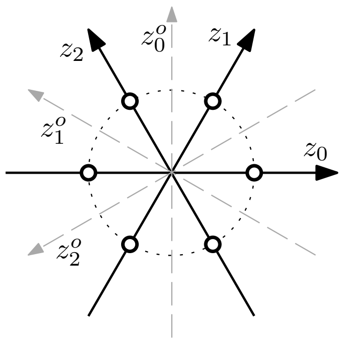


Figure 2.2: Additional drawing dimensions z_0, z_1 and z_2 for the iso-hexagon cartograms. As can be seen in the figure, the orthogonal directions must be explicitly expressed as none of the three dimensions are orthogonal to each other.

coordinate system, in case of the Demers cartograms or to a new set of directions z_0, z_1 and z_2, in case of the iso-hexagon cartograms. These directions are defined as follows. The z_0-dimension is the same as the x-dimension. Similarly to the y-dimension which is placed at a 90° rotational offset to the x-dimension, the z_1-dimension is placed at a 60° offset to the z_0-dimension and the z_2-dimension is placed at a 120° offset to the z_0-dimension. We will denote the coordinate of a point p in drawing dimension d as $d(p)$.

We will also talk about the orthogonal directions to these dimensions. Note that in case of the Demers cartograms, the y-direction is by definition orthogonal to the x-direction and vice versa. This is not true for the z-directions, as can be seen in Figure 2.2. Therefore we introduce a set of direction z_0^o, z_1^o and z_2^o, s.t., z_i^o is orthogonal to z_i and vice versa.

A line l is called *orthogonal to a dimension* d if all points on l have the same d-coordinate, i.e., $\forall p_l, p_l' \in l : d(p_l) = d(p_l')$. Similarly we define a line being parallel to a drawing dimension d.

We will use the drawing dimensions to specify locations of points. The d-coordinate $d(p)$ of a point p is determined by drawing a line through p orthogonal to d. All points on this line, including p, have by definition the same d-coordinate.

Note that in contrast to the x- and y-dimensions, the z-dimensions are linearly dependent, i.e., we can define a point p by only two z-coordinates which fixes the third one. Moreover we can and will express all z-coordinates of a point as a combination of the x- and y-coordinates of that point. These drawing dimensions are introduced to enable us to argue more intuitively about the directions in an iso-hexagon cartogram.

CHAPTER 3

Problem Description and Complexity

In this chapter we want to identify the problem statements relevant to this thesis and define them formally. We will also present two reductions that prove the NP-hardness of some of the defined problems.

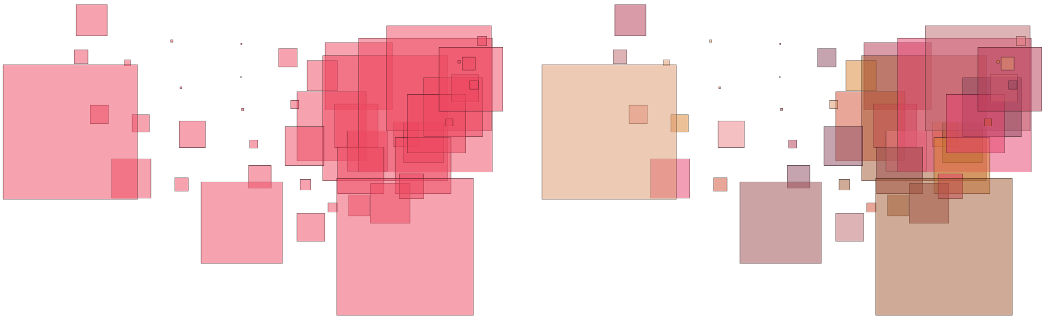
3.1 Drawing Demers Cartograms

For the purposes of this thesis we define a Demers cartogram as set of non-rotated interior-disjoint squares $\mathcal{R}$, a placement function $pos : \mathcal{R} \to \mathbb{R}^2$, which specifies the placement of the center of a square and a weight function $w : \mathcal{R} \to \mathbb{R}$ that specifies the desired area of a square. A iso-hexagon cartogram is defined similarly, but $\mathcal{R}$ is now a set of non-rotated interior-disjoint hexagons. Interior-disjointness is enforced, since with increasing overlap of regions, regions might completely obfuscated. Even if lower opacity values are used, i.e., the regions are to a degree transparent, they can become hard to differentiate. Examples of this can be seen in Figure 3.1. We will use squares/hexagons and regions interchangeably to refer to the squares or hexagons in a Demers cartogram or iso-hexagon cartogram.

3.1.1 Directional Relations

Preserving relational placement of regions inside a cartogram can be enforced as an additional requirement for a Demers cartogram or a iso-hexagon cartogram.

If a country lies in reality to the west of another, but the corresponding square of that country is placed to the left, i.e., east of the square of the second country, then this might be very confusing and hindering in finding and comparing countries in the cartogram. An example of this is displayed in Figure 3.2.



(a) (b)

Figure 3.1: Overlapping regions can obfuscate the cartogram such that it is difficult to differentiate between countries. (a) Drawing partly transparent regions does not solve this problem sufficiently and neither does (b) adding different colours to the regions.

In order to avoid this problem, we will identify so called *separation constraints*. A separation constraint is a set of pairs of regions. Depending on the type of cartogram we can identify either two (Demers cartogram) or three (iso-hexagon cartogram) separation constraints. This is dependent on the number of drawing dimensions which are available for that type of cartogram. For a Demers cartogram we define the separation constraints S_x and S_y, with the associated drawing dimensions x and y respectively. In a iso-hexagon cartogram the separation constraints are S_0, S_1, S_2, again with their associated drawing dimensions z_0, z_1 and z_2 respectively. If a pair (r_1, r_2) of two regions is element of a separation constraint S, we require this separation constraint to be fulfilled. This means that they must be separated in the drawing dimension d, which is associated with the constraint, i.e., we can draw a line l orthogonal to the dimension d, s.t., l is disjoint from the interiors of both r_1/r_2 and r_1 and r_2 lie on different sides of that line. Note again that l can coincide with the boundary of the squares of r_1 and r_2 and actually has to coincide with both in order to allow r_1 and r_2 to be adjacent. We call l the separating line of r_1 and r_2.

Note that the perceived relative direction of countries on a map is non-trivial. There are multiple methods of determining this relationship. Buchin et al [BKS+11] compare different methods of determining the relative placement of two regions and propose a new *splitting line model*. The method used by us is called the *centroids model*. We chose this model despite the results of Buchin et al. due to the fact that separation constraints must be symmetrical, which their splitting line model is not. We will explain this model in detail in Chapter 4.

We can define a *strong setting*, in which we add a pair of regions to both separation constraints, based on their properties in the input. If not otherwise specified we will however assume, that we are talking about the *weak setting*, in which every pair of regions is in exactly one separation constraint. “Two regions are in a separation constraint” and “Two regions have a separation constraint” are used interchangeably.

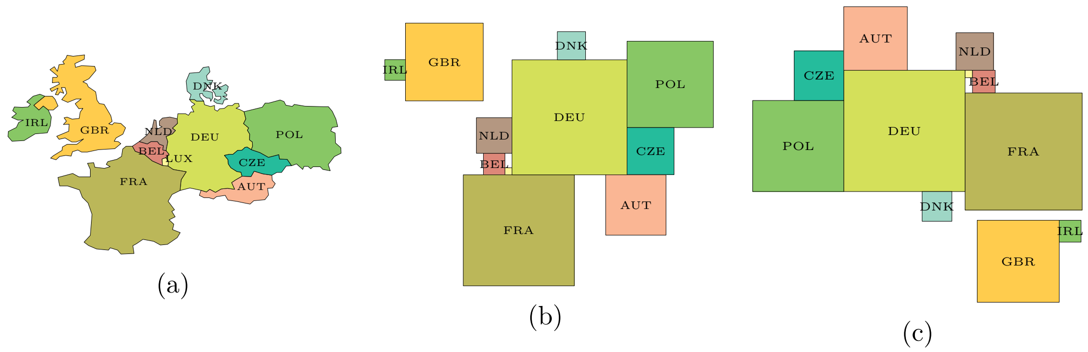


Figure 3.2: Directional relations should be kept in a cartogram, (a) is an input map in which we can identify directional relations like "Poland is to the right of Germany", (b) is a Demers cartogram which keeps these directional relations (in general), while (c) violates a number of them and leads to a worse cartogram. In the cartograms, Luxembourg is not labeled due to space constraints.

3.1.2 Desired Properties of a Good Cartogram

In this section we want to list the quality criteria, which are desired in a "good" cartogram. In a practical application, these are the properties, which are maximized. These properties can however stand in conflict with each other. It is therefore necessary to choose weights that relate the properties to each other and make them comparable.

The first property is, that areas which are adjacent are kept adjacent and those that are not, are not placed next to each other in a cartogram. We call the first "keeping adjacencies" and the second "not creating new adjacencies". An easy real life example that illustrates that this constraint is not trivial is the placement of the countries Brazil, Argentina, Bolivia and Paraguay as illustrated in Figure 3.3.

This constraint is something that can be fulfilled partially, but it is worth mentioning that the number of lost adjacencies can obviously only ever be an integer number. This is relevant in for reasons mentioned in Section 2.2. The goal behind keeping adjacencies and not creating new adjacencies is to create analogies between the input map and the cartogram that aid the user in locating and identifying countries in the cartogram.

By ensuring the separation constraints are kept, we already restrict the placement of squares relative to each other. However, since we identify only one direction in which we guarantee the separation constraint, e.g., region a must be placed completely to the right of region b, the y-coordinate of b is not fixed and therefore the region can be placed anywhere from the lower to the upper boundary of the drawing area. One method to fix this is to try to keep the slope of the line through the two centers of a and b as close as possible to the slope of the line through their centroids in the input.

It can also be helpful to keep the original position of regions in the cartogram. This is achieved in minimizing the distance of the center point of a square to the position of the

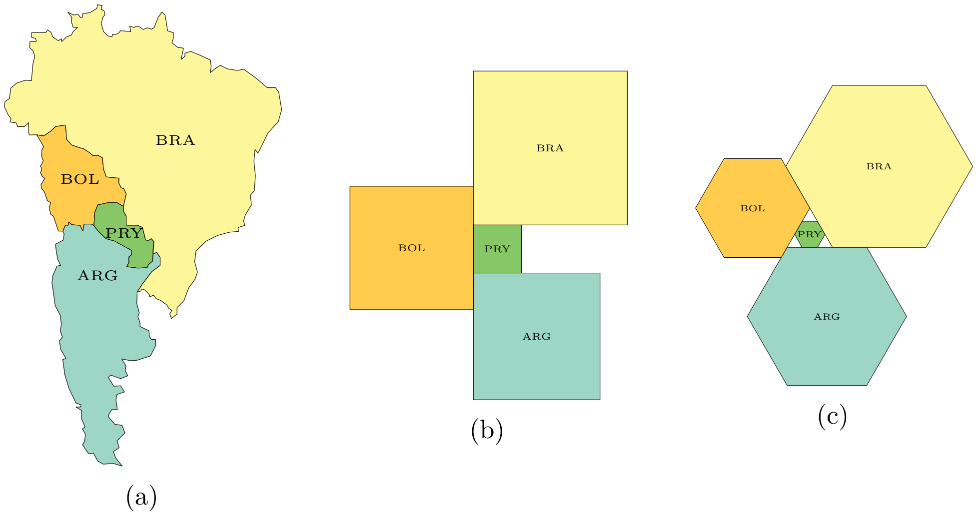


Figure 3.3: Regardless of the sizes assigned to the countries in the Demers cartogram (b), the adjacencies defined in the input map (a) cannot be kept. Note that, for this configuration, this is not true in a iso-hexagon cartogram (c).

centroid in the input. This tries to keep the original shape of the map. The distances can be measured with the metrics described in Section 2.3.

3.1.3 Formal Description

We are now prepared to present formal problem description of the Demers cartogram problems. We will present two different problems in this section. One is a question of existence, i.e., is there a Demers cartogram for a set of regions that keeps all adjacencies without creating new ones and the other is minimization problem in which we try to minimize the number of lost adjacencies.

Problem 1 (Demers Cartogram Existence Problem)**.** *Given an adjacency Graph G with a planar embedding and separation constraints S_x, S_y and a weight function w, does there exist a Demers cartogram which draws all regions $r \in \mathcal{R}$ as squares of size $w(r)$, keeps all adjacencies and fulfills all separation constraints?*

The adaption of this problem into a minimization problem is done by allowing for lost adjacencies, but adding them as an optimization goal. The formal description of this problem is as follows.

Problem 2 (Demers Cartogram Minimization Problem)**.** *Given an adjacency Graph G with a planar embedding and separation constraints S_x, S_y, and a weight function w_i. What is the minimal number of lost adjacencies over all possible Demers cartograms, which draws all regions $r \in \mathcal{R}$ as squares of size $w(r)$ and fulfills all separation constraints?*

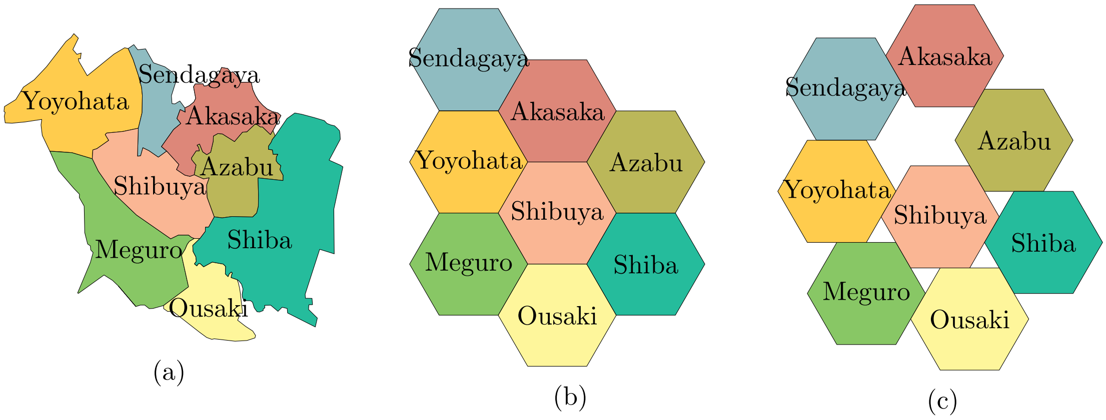


Figure 3.4: (a) Input map of Tokyos municipality Shibuya and its 7 surrounding municipalities. Assuming all regions have equal size, this map can not be turned into a iso-hexagon cartogram without losing at least a single adjacency, neither with a dense packing (b) nor by loosening the packing (c).

3.2 Drawing iso-hexagon cartograms

A iso-hexagon cartogram is an adaptation of a Demers cartogram. Instead of representing every region with a square, we represent them with regular hexagons. Most of the concepts of the Demers cartograms are directly applicable to the iso-hexagon cartograms.

Problem 3 (Iso-Hexagon Existence Problem)**.** *Given an adjacency Graph G with a planar embedding and separation constraints S_0, S_1 and S_2 and a weight function w, does there exist a iso-hexagon cartogram which draws all regions $r \in \mathcal{R}$ as regular hexagons of size $w(r)$, keeps all adjacencies and fulfills all separation constraints?*

Problem 4 (Iso-Hexagon Minimization Problem)**.** *Given an adjacency Graph G with a planar embedding and separation constraints S_0, S_1 and S_2 and a weight function w. What is the minimal number of lost adjacencies over all possible iso-hexagon cartograms, which draws all regions $r \in \mathcal{R}$ as regular hexagons of size $w(r)$ and fulfills all separation constraints?*

3.3 Leader Lines

Let G be the adjacency graph of the input map. G has a planar embedding. As already mentioned, even with a planar embedding given, we might not be able to keep all adjacencies of the input, if all regions have fixed size. Examples for necessarily lost adjacencies can be seen in Figure 3.3b for the Demers cartograms and for a iso-hexagon cartogram in Figure 3.4.

These lost adjacencies are nevertheless informative to the user and we can try to mitigate the impact of lost adjacencies by visualizing them through curves connecting the two regions. We will call such a curve, a *leader line* and we will place specific restrictions on them in order to integrate them into the specific design of the cartogram. Relating to these leader lines we can pose an additional problem.

Problem 5 (Leader Line Existence Problem)**.** *Given an adjacency Graph G with a planar embedding and separation constraints S_0, S_1 and S_2 and a weight function w. Assuming that there exists a hypothetical Cartogram $\mathcal{A}$ in which all regions are realized with a polygon of correct shape and arbitrary size and all adjacencies and separation constraints are fulfilled, can we always find a Cartogram $\mathcal{B}$ in which all regions $r \in \mathcal{R}$ are represented by polygons of correct shape and and area of $w(r)$ and all adjacencies can be visualized with a leader line of bounded length, such that, this line and all regions except the ones it is connecting are disjoint?*

We will give a proof of the existence of such a leader line and a bound in relation to the placement of the regions the line is connecting in Chapter 5

3.4 Hardness

In this chapter we want to prove the NP-completeness of a generalized versions of the problems 1, 2, 3 and 4. For the first two, we will first pose a rephrased version of the problems, and then follow a chain of results already proven in other papers to establish NP-completeness. For the iso-hexagon cartogram problems, we will give a reduction that mimics a similar reduction already used in the other papers.

3.4.1 Generalization

A Demers cartogram without lost adjacencies is clearly a square contact representation of the adjacency graph. However, as already mentioned, due to the fixed sizes of the regions, the existence problems are not equivalent. If we set all weights of the regions to the same value and drop the necessity of the separation constraint we can state the following generalized problem.

Problem 6 (Proper Square Contact Representation)**.** *Given a Graph $G = (V, E)$, is there a set of unit squares S, s.t., there is a bijective function $f : V \to S$, all pairs of squares $f(u) = a, f(v) = b$ are in contact if and only if a and b share a boundary segment of non-zero length?*

The same can be done for the iso-hexagon cartogram and the proper hexagon contact representation. However we need to augment the problem instance to include a weight function as the proof of NP-hardness requires hexagons of fixed but different sizes, but we are able to drop the necessity for a proper contact representation.

Problem 7 (Hexagon Contact Representation)**.** *Given a Graph $G = (V, E)$ and a function $w : V \to \mathbb{R}$, is there a set of hexagons H, s.t., there is a bijective function*

$f : V \to H$, s.t., the size of hexagon $f(v)$ is exactly $w(v)$ and all pairs of hexagons $f(u) = a, f(v) = b$ are in contact if and only if $\{u, v\} \in E$?

The minimization problems are similarly defined, but allow for lost adjacencies. The problem then asks for the minimal possible number of lost adjacencies.

3.4.2 NP-Completeness of Proper Square Contact Existence

Problem 6 has been shown to be NP-complete by Kleist and Rahman [KR14], as a secondary result in a paper about cube representation of graphs. They achieve this by building a logic engine with unit squares similar to the ideas of [BEF$^+$12] and with that reducing Not All Equal 3-Sat to the proper square contact representation problem.

Theorem 3.4.1 (Consequence of [KR14])**.** *The Proper Square Contact Existence problem is NP-hard.*

3.4.3 NP-hardness of Proper Square Contact Minimization

Theorem 3.4.2. *The Square Contact Minimization problem is NP-hard.*

Proof. We will show NP-Hardness for the corresponding minimization problem, by reducing the existence problem to the minimization problem. This is done straightforward. Given an instance of Problem 6 we can use it as is as the instance of the minimization problem. Assume there exists a polynomial time algorithm for the square contact minimization problem, then we could simply solve the instance. If the resulting answer is 0 we return true, otherwise we return false, since a minimal number of lost adjacencies bigger than zero makes the existence of a proper square contact representation without lost adjacencies impossible. Since Problem 6 is NP-complete, we know that the minimization problem is NP-hard. □

3.4.4 NP-Hardness of Hexagon Contact Existence

We replicate the construction of a logic engine as done by Kleist and Rahman [KR14], however instead of unit-squares, we will use hexagons of varying but fixed sizes as building blocks. We will first present the Not All Equal 3-Sat problem, then explain the underlying structure of the construction, we will explain how all parts of this structure are replicated using hexagons, discuss why the construction is rigid (for all intents and purposes of this problem) and finally we will give a high level overview of the complete reduction.

Not All Equal 3-Sat

Sat is famous for being the first problem for which NP-completeness was proven by Cook [Coo71]. It is the problem of finding a satisfying variable assignment for a given boolean formula φ. It is also a popular starting point for reductions to other problems, since there exist various restrictions to the Sat problem which remain NP-complete.

One of these adaptions is Not All Equal 3-Sat or NAE3Sat. An instance of NAE3Sat consists of a boolean formula φ in 3-CNF Form. A formula φ is in 3-CNF Form, if it consist of a set of clauses, which are all connected by a $\wedge$-operator and every clause consists of at most 3 literals which are connected by a $\vee$-operator. The formula φ in 3-CNF Form is a positive NAE3Sat instance, if and only if there exists a variable assignment, which makes φ true and every clause contains at least one literal which is false under the variable assignment.

The formal problem description of the NAE3Sat problem is given below.

Problem 8 (Not All Equal 3-Sat)**.** *Let $\varphi = (\mathcal{V}, \mathcal{C})$ be a formula, $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$ be the set of all variables in φ, $\mathcal{C} = \{c_1, c_2, \dots, v_n\}$ be the set of all clauses in φ and let a clause $c \in \mathcal{C}$ be a set of variables, s.t. $\forall c \in \mathcal{C} : |c| \leq 3$. Decide if there exists a variable assignment I, s.t.,*

$$I(\varphi) = true$$

$$\forall c \in \mathcal{C} \exists v \in c : \begin{cases} I(v) = false & if \quad v \in c \\ I(v) = true & if \quad \neg v \in c \end{cases}$$

The NP-completeness of this problem can easily be shown by reduction from 3-Sat and was used by Bremner et al. [BEF$^+$12] and by extension Kleist and Rahman [KR14].

Proof using a Logic Engine

Theorem 3.4.3. *The Hexagon Contact Existence problem is NP-hard.*

Proof. A logic engine is a graph, in which we can map certain parts of the graph to components of a logical formula. The construction of the logic engine directly follows the description by Bremner et al. [BEF$^+$12]. The engine consists of a central horizontal spine. For every variable in $\mathcal{V}$, we will attach two poles to this central spine, one above and one below the *spine*. All n *poles* are placed from left to right in equal distances. Every pole consists of m connection points, one for each clause, which we call *clause components*. The entirety of this logic engine is then surrounded by a frame, which is multiple vertices thick in order to make it impossible for parts of the frame to shift independently of the rest of the frame.

We initially place $n + 1$ non-connected vertices from left to right on a horizontal line. These vertices will build the spine. The poles consist of m copies of the three vertices p_1, p_2, p_3, s.t., $p_1 - p_2$ and $p_2 - p_3$ are connected. These vertices are a clause component. The p_3 of every clause component is connected to the p_1 of the next. We build two such poles for every variable. We will call one the *positive pole* and the other the *negative pole*. The j-th poles, i.e., the poles for variable v_j are placed, s.t., the p_1 of their first components are connected to the j-th and $(j+1)$-th vertex of the spine, as well as to each other. By placing n pairs of poles, we connect all vertices of the spine. Furthermore every pole can be drawn above or below the spine, independently of the other poles (except its opposite pole).

The *frame* consists of 6 chains of vertices placed above, below, to the left and to the right of the already constructed parts. The two chains, placed above and below, consist of $3 \cdot (2n+3)$ vertices, while the four chains on the sides consist of $3 \cdot (3m)$ vertices (which is the same number of vertices as in a negative or positive pole for a variable). The left and right ends of the chains above and below are connected each to one end of a chain on the side with 4 additional vertices. The other ends of the two chains on the left are connected to the first vertex of the spine and conversely the other two chains on the right are similarly connected to the last vertex of the spine. Additionally we connect, the two side chains on the same side with one additional vertex, each. The entirety of the construction up to this point is depicted in Figure 3.5a on the next page.

The central idea of this construction is that every pair of poles, represents one variable. One pole represents the variable as a positive literal while the other represents the variable as a negative literal. For a variable v_i we will call the first pole o_i^+ and the second o_i^-. Let t^k be the k-th clause component in a pole. The set of all k-th clause components over all poles represents the k-th clause of the formula. If and only if a variable v_i does not occur as a positive literal in clause k, we attach a flag vertex to vertices p_2 and p_3 of the k-th clause component of o_i^+. Conversely we attach a flag vertex to vertices p_2 and p_3 of the k-th clause component of o_i^- if and only if v_i does not occur as a negative literal in clause k.

A variable v_i is thought of as true, if o_i^+ is place above the spine and therefore above o_i^-, otherwise its thought of as false. Let p_f, p'_f be flag vertices adjacent to the k-th clause component of two adjacent poles. The size of the hexagons representing p_f and p'_f is chosen in such a way, that only one such hexagon can ever be drawn simultaneously in the space between the two adjacent poles. The space between the frame and the outer poles will be restricted , s.t., no flag vertex can be drawn in this space. This means that we need at least one k-th clause component over all poles above the spine without a flag vertex attached, in order to have enough space to draw all flag vertices on that level. Conversely we need a similar clause component for every row of clause components below the spine. Therefore we can relate a clause component above the spine without a flag vertex—which enables the k-th level of clause components above the spine to draw their flag vertices—to a variable which makes the k-th clause true. This can either mean, the variable itself is true and its positive pole is drawn above the spine or the variable is false and its negative pole is drawn above the spine. In the same way tje k-th clause component without a flag vertex below the spine can be related to a variable that is false in the k-th clause.

At this point the central idea of the reduction is complete and we know that every formula that can be fulfilled can be drawn as a hexagon contact representation with the given weights. There are however still drawings of graphs that relate to formulas which can

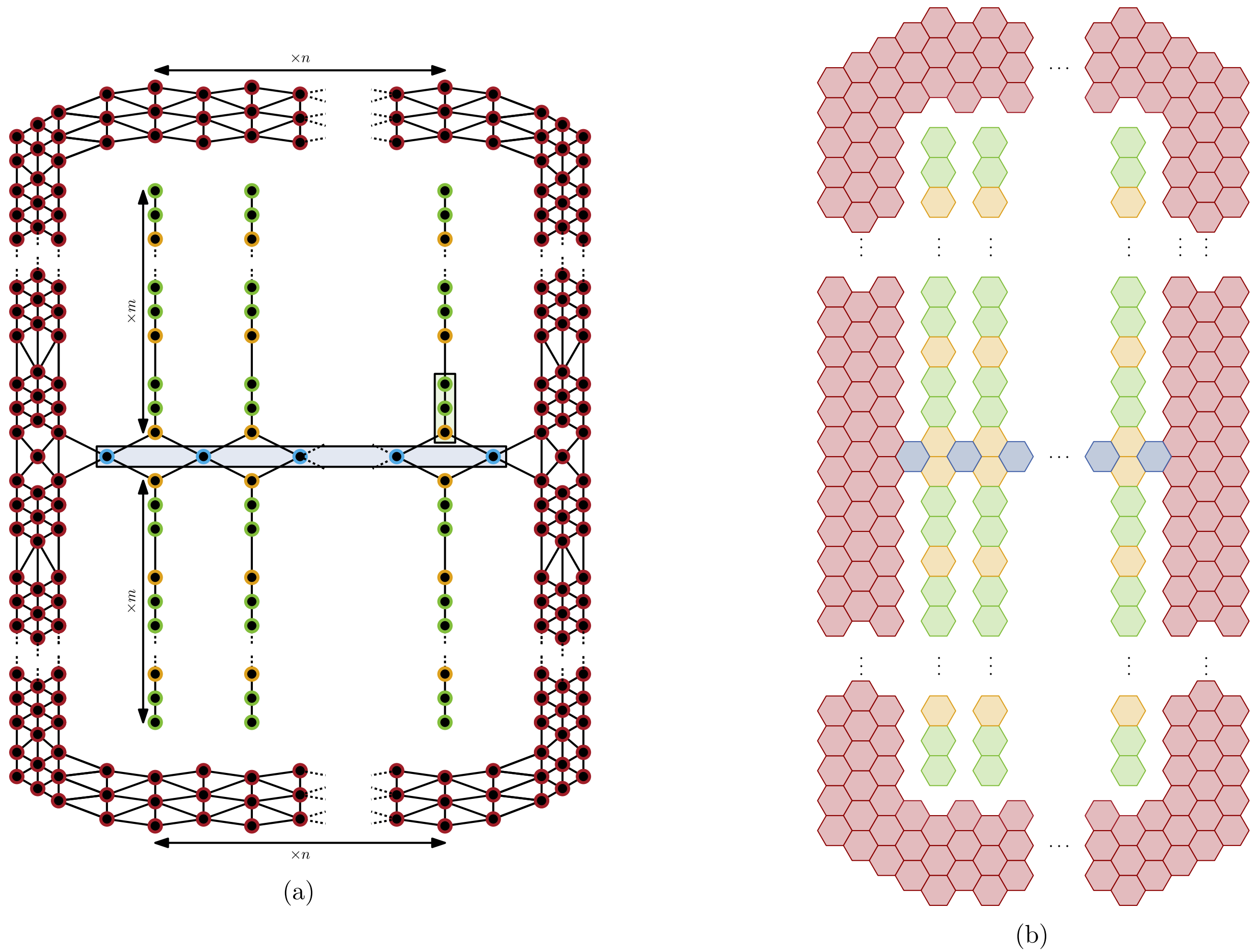


Figure 3.5: Construction of the basic structure of the logic engine. (a) shows the construction of the underlying graph. (b) shows an example realization of the underlying graph as a proper hexagon contact representation

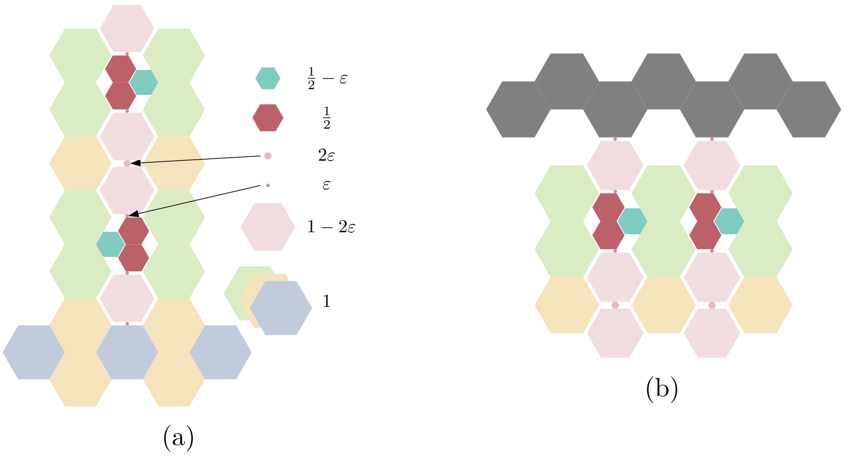


Figure 3.6: Detailed construction of the spacer chains. The spacer chains are connected to the spine and consist of hexagons of 4 different sizes. The first are hexagons of height $1 - 2\varepsilon$. They are connected to the spine (a) and the frame (b) with small hexagons of height ε. Two of these big hexagons are connected by a small one of height 2ε. The empty pockets between the poles in which a flag hexagon might be drawn, contain two hexagons of height $\frac{1}{2}$, which are connected to a big hexagon by a small one of size ε.

not be fulfilled, due to the fact that the poles are not forced to be drawn in a straight line. More precisely, we want to guarantee that in a pole, a hexagon is connected to the previous and the successive hexagon through opposing sides and that we are not able to move hexagons so far along these edges that we create enough space between the poles for two flag variables to be drawn on one level.

To guarantee this, we want to make this construction rigid, in a sense. We will place chains of vertices between the poles (so called *spacer chains*), which are not adjacent to the poles but only to the spine in order to retain the possibility to flip the poles independently from each other to the other side of the spine. Further we will scale all components of these spacer chains, s.t. the distance between the components and the poles is minimal, in order to limit movement. The composition of such a spacer chain and the relative sizes of all components of the spacer chain are displayed in Figure 3.6a.

The spacer chains are connected at the top to the lower points of the top end of the frame, as shown in Figure 3.6b. The two smaller hexagons of height $\frac{1}{2}$ can be drawn simultaneously to a hexagon of a flag vertex in their pocket, but not simultaneously to two flag vertex hexagons in the same pocket. At the same time we can not draw the two hexagons of height $\frac{1}{2}$ and the next hexagon of size $1 - \varepsilon$ in the same pocket, since ε is chosen as small as necessary to make this impossible. The space between the frame and the outer poles is filled up as shown in the example in Figure 3.7, blocking all flags from being placed outside the outer poles. This also entails that we can never use an empty pocket in a level of clause components to draw parts of a pole into it, in order to make space in a level further up the pole, making it in turn impossible to draw a level

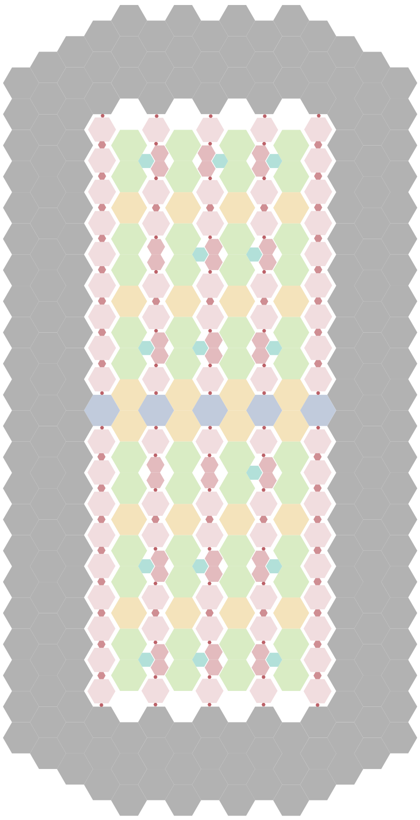


Figure 3.7: Complete construction for formula φ, including the correctly placed flag hexagons in dark green. The construction is rotated to the right, i.e., the right side is the top.

of clause components which all have a flag vertex attached, by changing their height in the drawing. The final construction including all spacer chains and all flag hexagons for the satisfiable formula φ (defined below) is shown in Figure 3.7. An exemplary variable assignment $\mathcal{I}$ that satisfies φ is $\mathcal{I}(x_1) = f, \mathcal{I}(x_2) = f, \mathcal{I}(x_1) = f, \mathcal{I}(x_4) = t$.

$$\varphi = (x_1 \lor x_2 \lor \neg x_3) \land (x_4 \lor x_3 \lor \neg x_1) \land (\neg x_4 \lor x_3 \lor \neg x_2)$$

Assume, that a formula φ has a variable assignment $\mathcal{I}$ which fulfills φ, s.t., every clause contains a literal, which is false. Then a drawing of the constructed graph exists, if the poles are rotated in such a way that o_i^+ is drawn above the spine if v_i is true under $\mathcal{I}$ and below the spine otherwise, in which case o_i^- is drawn above the spine. Since φ is satisfied under $\mathcal{I}$, the k-th clause contains a literal which is true. If this literal is positive, the variable if true, o_i^+ is drawn above the spine and the k-th clause component does not have a flag vertex attached which enables all k-th clause components above the spine to draw their flag vertices. This holds for all $k \in [1, m]$. Since every clause also contains a literal which is false, this also holds for all poles below the spine.

Assume now, that the constructed graph has a hexagon contact representation with the given fixed sizes. Then every pole could draw every connected flag vertex as a hexagon without overlap or creating new adjacencies. The existence of the spacer chains introduced hexagons between the poles, which are not adjacent to the poles but have a very small distance of ε. This forces all poles to be drawn in such a way, that all hexagons of the pole are connected to each other through their opposite sides. We can shuffle the hexagons minimally to the right and left, but the maximal displacement between one hexagon and the next is smaller than $4(n+1) \cdot \varepsilon$, since we can at most shuffle one row of hexagons completely to the left, and the next completely to the right by at most the distance between a spacer chains biggest component and the pole times the number of spaces between the spacer chains and the poles (including the space between the spacer chain and the frame). Since we need to shift two hexagons with an edge length of 1 at least $\frac{1}{2}$ to the side, in order to create a connection through a different edge, we can clearly chose ε small enough to prevent this. This means that all k-th clause components are drawn at the same height. From that we can follow that at least one of the k-th clause components above the spine and at least one below the spine do not have a flag hexagon, since that would make a drawing impossible. Let the pole of the clause component, which enables this above the spine be the i-th pole. If this pole is o_i^+ we set v_i to true, otherwise we set it to false. This results in a satisfying variable assignment since the poles o_i^+/o_i^- are drawn either completely above or completely below the spine. This concludes the proof. □

3.4.5 NP-hardness of Hexagon Contact Minimization

Theorem 3.4.4. *Hexagon Contact Minimization is NP-hard.*

Proof. This proof follows the same argument as the proof for the proper square contact minimization. We again use the same instance, run an algorithm, which solves the hexagon contact minimization and return true if the answer is zero and false otherwise. Due to the proof above, hexagon contact minimization is NP-hard. □

3.5 Quality Measurements

We want to be able to judge a cartogram or a set of cartograms. For that we will use different measurements, some of which were developed for treemaps. The similarities between treemaps and cartograms were pointed out in Chapter 1.1.

Nusrat et al. [NK16] identified three general quality criteria for a cartogram, namely *statistical*, *geographical* and *topological accuracy*. We present our measures for the applicable quality criteria, as well as measures for an additional criteria we call *stability*, which tries to capture how two cartograms relate to each other.

3.5.1 Statistical Accuracy

Statistical accuracy measures the difference between the size a representation of a region should have in a cartogram and its actual size. The difference is commonly called *cartographic error* and is by definition of the Demers cartogram and iso-hexagon cartogram not a relevant factor since all regions are represented by a polygon of exact and fixed size.

3.5.2 Geographical Accuracy

Geographical accuracy measures how closely the cartogram resembles the input map. This is judged by how much the shape and position of a region in the cartogram resembles the shape and position in the input. The first is not applicable in this setting since the shape of a region is fixed and cannot differ. The second can be measured as the average displacement of a region from its origin. Let $o(r_0)$ be the original position of a region in the input and let $o'(r_0)$ be the center of the polygon of the region in cartogram P. Then the *origin displacement* δ_o is defined as

$$\delta_o = \frac{|o(r_0) - o'(r_0)|}{C_{BB}}$$

This measure can clearly be used for both the Demers cartograms and the iso-hexagon cartograms. C_{BB} is the maximum over the length of the two diagonals of the bounding boxes of the input and P. We divide δ_o by this large constant to obtain a value between 0 and 1, since the full diagonal of the bigger layout is the maximal distance, a square could travel. In practical applications, the values tend to range between 0 and 0.3.

3.5.3 Topological accuracy

Topological accuracy captures how well the input topology is represented in the cartogram. This relates to the adjacencies between regions and will be measured by the number of supposedly adjacent regions which are not actually adjacent in the Demers cartogram or iso-hexagon cartogram. This calculation of lost adjacencies is done after creating the cartogram by using the following formula for two squares s_1, s_2.

$$adj(s_1, s_2) = \begin{cases} true & \text{if} \max(\delta_x, \delta_y) < \frac{w(s_1)+w(s_2)}{2} \\ false & \text{otherwise} \end{cases}$$

The measurement can easily be adapted to two hexagons h_1, h_2.

$$adj(h_1, h_2) = \begin{cases} true & \text{if} \max(\delta_0, \delta_1, \delta_2) < \frac{w(h_1)+w(h_2)}{2} \\ false & \text{otherwise} \end{cases}$$

The δ_d variables are the distances between the centers of two polygons in the relevant drawing dimension d. Note that we do not measure the number of newly created adjacencies, as our method does not allow for that to happen.

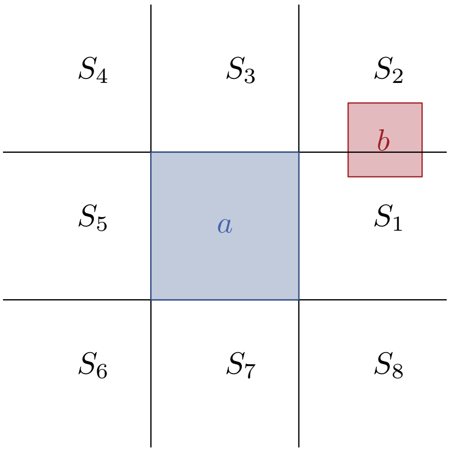


Figure 3.8: One third of b is in Sector S_1 of a, two thirds in S_2.

3.5.4 Stability

Additionally we want to measure not only the quality of a single cartogram, but of a set/series of cartograms and how stable two cartograms are in relation to each other. Stability of such a pair tries to capture how much the mental model of the user is disturbed, when switching from one cartogram to the other. Small changes in the input should result in small changes in the output and vice versa. Note that due to changing data values the size and position of a region is almost always forced to change from one cartogram to the next, however some movements are considered to be worse than others, i.e., moving a region b which was to the right of region a slightly more to the right can be expected to keep the mental image of the user better than flipping b to the left side of a.

We will employ different measures to capture stability. The first is *stability score* of Sondag et al. [SSV18]. The score splits the surrounding area of a rectangle a into 8 sectors, defined by the lines going through its sides. It then measures the fraction of a second region b that is in each sector of a. Figure 3.8 shows an example of that. The result is an 8-dimensional vector $\beta_{ab} = [\beta^1_{ab}, \ldots, \beta^8_{ab}]$, s.t., β^i_{ab} is equal to the fraction of b that is in S_i of a. Let a, b be the squares of two regions r_a, r_b in a Demers cartogram P and a', b' be the squares of r_a, r_b in a second Demers cartogram P'. The relative position change of r_a, r_b from P to P' is calculated as

$$C_{PP'}(r_a, r_b) = \frac{1}{2} \cdot ||\beta_{ab} - \beta_{a'b'}||_1$$

The stability score of two cartograms P and P' is defined as

$$S_{PP'} = \frac{\sum_{i \neq j} C_{PP'}(r_i, r_j)}{n-1}$$

In a iso-hexagon cartogram we need to redefine this measure. Using a similar definition of dividing the surrounding area of a hexagon by drawing lines through the edges of said

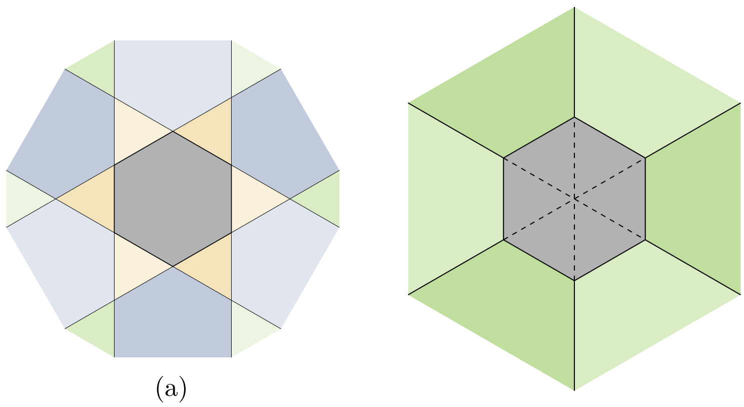


Figure 3.9: Adaption of the stability score for a iso-hexagon cartogram. (a) Defining the sectors with lines through the edges of a hexagon leads to unintuitive sectors of unfavorable size and shape. (b) Sectors are defined by lines through opposing corners.

hexagon result in overlapping sectors of unfavorable size and shape (see Figure 3.9a). Instead we use lines through opposing corners of a hexagon to define the sectors, as in Figure 3.9b. This leads to only 6 sectors instead of 8, which results in a 6-dimensional vector β_{hex}. With this vector we calculate the stability score as above.

Note that this first measure is the only one, that captures relative placement of regions between cartograms, while all following measures only track the movement of a single region between cartograms.

The second criteria measures the distance between the center of a square of a region r in two different cartograms P, P'. This measure δ_{ct} will be called *center movement distance*. Let c, c' be the centers of the polygons in cartograms P, P' respectively. Then δ_{ct} is defined as:

$$\delta_{ct} = \frac{|c - c'|}{C'_{BB}}$$

This measure can clearly be used for both Demers cartograms and iso-hexagon cartograms. Similar to δ_o we use C'_{BB} to scale this measure to a range of $[0, 1]$. However we now take the maximal length of a bounding box diagonal over the two layouts P, P'.

The third measure keeps track of the corner movement of a region between cartograms. The *corner travel distance* δ_{co} is defined as the sum of all distances between corners. Let the squares of region r_a in the Demers cartograms P, P' be given by their corners (c_1, c_2, c_3, c_4) and (c'_1, c'_2, c'_3, c'_4) respectively. Then δ_{co} is defined as:

$$\delta_{co} = \frac{|c_1 - c'_1| + |c_2 - c'_2| + |c_3 - c'_3| + |c_4 - c'_4|}{4 \cdot C_{BB}}$$

We use the same large constant to scale this measure as for δ_{ct}, simply multiplied by four.

For usage in a iso-hexagon cartogram we need to adapt this measure to a new one called δ_{co}^{hex}. Let the hexagons of region r_a in P, P' be given by their corners $(c_1, c_2, c_3, c_4, c_5, c_6)$ and $(c'_1, c'_2, c'_3, c'_4, c'_5, c'_6)$ respectively. Then δ_{co}^{hex} is defined as:

$$\delta_{co}^{hex} = \frac{|c_1 - c'_1| + |c_2 - c'_2| + |c_3 - c'_3| + |c_4 - c'_4| + |c_5 - c'_5| + |c_6 - c'_6|}{6 \cdot C_{BB}}$$

Sondag et al. [SSV18] mention that this measure is related to the measure of Schneidermann and Wattenberg [SW01], however since that measure factors in the size change of a rectangle and the size changes between different cartograms are fixed in our scenario we discarded the Schneidermann and Wattenberg metric.

CHAPTER 4

Linear Program

A linear program is an approach to model a problem mathematically by introducing variables, placing constraints on those variables and then defining an objective function which is minimized. A problem is modeled such that, the constraints to enforce the desired properties of a good solution and a minimal value of the objective function corresponds to an optimal solution of the problem. When we find a variable assignment that minimizes the objective function, we can then extract the information about this optimal solution from the values of the variables. Linear programs or LP's are formally defined in 2.2.

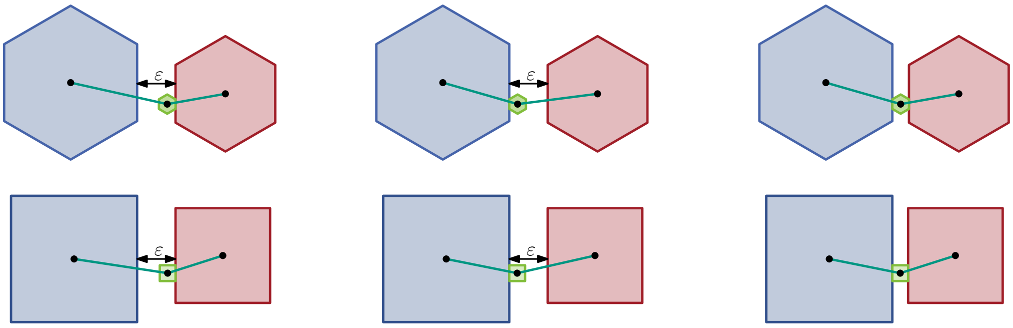
In the following, we will present the formalization of the Demers cartogram minimization problem (problem 2), which is based on the sketch of Chimani et al. [CKM+18], that resulted from the Shonan Meeting No.127 in 2018.

After that, we will present how this formalization is adapted to the hexagonal case using ideas, similar to Nickel and Nöllenburg [NN19].

4.1 LP-Model for Demers cartograms

Let $\mathcal{R}$ be the set of all regions, $r \in \mathcal{R}$ a single region and let w be a function which associates a real number with every region, s.t. r is displayed in Cartogram C by a square of area $w(r)$. The position of a region r in the input is given by its center $c(r) = (x(r), y(r))$. Adjacencies are given in input graph $G = (\mathcal{R}, \mathcal{T})$. The x- and y-coordinates of the regions in $\mathcal{R}$ define a planar embedding of G.

We will first explain how to formalize an LP, that creates a single Cartogram and we will extend this idea to the simultaneous or iterative creation of multiple cartograms afterwards.



(a) (b) (c)

Figure 4.1: In the possible configurations either the adjacency $\{r_0, r_2\}$ or $\{r_2, r_1\}$ is lost. However a configuration (c) exists in which all adjacencies are realized and all separation constraints are fulfilled.

4.1.1 Position of Regions

For every region $r \in \mathcal{R}$, we create a variable x_r, y_r that encodes the x- and y-coordinate of r. We will use the final values of these variables to extract the positioning of r in the resulting Demers cartogram.

4.1.2 Disjointedness and Separation Constraints

We now want to enforce non overlapping hexagons in the LP. For this constraint we differentiate between regions r_0, r_1 which were originally adjacent, i.e. $r_0, r_1 \in \mathcal{T}$ and ones which were not. For the former we define the variable $gap_{r_0,r_1} = 0$. For the latter we define $gap_{r_0,r_1} = \varepsilon$, where ε is a predefined value for a minimal distance of two adjacent regions. This value must be chosen with some care, since the existence of a region r with an associated value $\sqrt{w_i(r)} < \varepsilon$ can lead to an adjacency which can not be realized, solely on the account of the minimal distance of its neighbors, see fig 4.1. In the figure, we have $\{r_0, r_1\} \notin \mathcal{T} \implies gap_{r_0,r_1} = \varepsilon$. Since $\{r_2, r_0\} \in \mathcal{T}$ and $\{r_2, r_1\} \in \mathcal{T}$ we have $gap_{r_2,r_0} = gap_{r_2,r_1} = 0$. Because $\sqrt{w_i(r)} < \epsilon$, r_2 can only be placed adjacent to one of the two other regions, not both.

The minimal distance required between the centers of two regions r_0, r_1 for them to be non overlapping is

$$D_{01}^{min} = \frac{(w_i(r_0) + w_i(r_1))}{2}$$

To enforce this distance on the regions we add the following constraints into the LP.

$$x_{r_1} - x_{r_0} \geq D_{01}^{min} + gap_{r_0,r_1} \qquad \forall (r_0, r_1) \in S_x \qquad (4.1)$$

$$y_{r_1} - y_{r_0} \geq D_{01}^{min} + gap_{r_0,r_1} \qquad \forall (r_0, r_1) \in S_y \qquad (4.2)$$

In order to keep adjacent regions adjacent in the output, we want to add the following constraints. The variables $s_x^{r_0,r_1}, s_y^{r_0,r_1}$ measure the distance between r_0 and r_1.

$$s_x^{r_0,r_1} \geq \max((x_{r_0} - x_{r_1}) - D_{01}^{min}, (x_{r_1} - x_{r_0}) - D_{01}^{min}) \qquad \forall\{r_0, r_1\} \in \mathcal{T} \tag{4.3}$$
$$s_y^{r_0,r_1} \geq \max((y_{r_0} - y_{r_1}) - D_{01}^{min}, (y_{r_1} - y_{r_0}) - D_{01}^{min}) \qquad \forall\{r_0, r_1\} \in \mathcal{T} \tag{4.4}$$
$$s_x^{r_0,r_1}, s_y^{r_0,r_1} \geq 0 \qquad \forall\{r_0, r_1\} \in \mathcal{T} \tag{4.5}$$

Now we can minimize these distances.

$$\min \sum_{\{r_0,r_1\}\in\mathcal{T}} s_x^{r_0,r_1} + s_y^{r_0,r_1} \tag{4.6}$$

Since we can weigh this constraint to such an extend, that all other constraints are dominated by this one, we know that our linear program model finds a cartogram in which we do not lose any adjacencies, if that is possible with the given weights. Note however, that, if this is not possible, there are situations in which the LP does not find the minimal number of lost adjacencies. As the LP only minimizes the sum over all distances d, it can not differentiate between a situation where $s_x^{r,r'} = 0$ and $s_x^{r,r''} = 6$ and another situation in which $s_x^{r,r'} = s_x^{r,r'}, = 3$ even though it is obvious that the number of lost adjacencies in the first case is one, while it is two in the second.

Strong Setting

For the strong setting we need to instantiate another separation constraint if we have an additional separation in the input. Let r_0, r_1 be two regions, s.t., $(r_0, r_1) \in S_x$ and r_0, r_1 have a horizontal separating line in the input with r_0 being placed completely below the line and r_1 completely above. Then we add the constraints 4.2 additionally to ensure the separation in the second dimension. All other cases can be handled similarly.

4.1.3 Slope

A result from the formulation of these constraints is that the two variables $s_x^{r_0,r_1}, s_y^{r_0,r_1}$ both have value zero if and only if the squares of r_0 and r_1 are touching. However this leads to multiple optimal solution. As long as separation constraints are kept, the representational squares for two regions can slide along their touching sides, without changing the optimal value and therefore without multiple placements are considered optimal by the LP.

In order to determine a single optimal relative placement for two regions in an output drawing, we add a secondary constraint, which tries to keep the slope between the two regions from the input intact. This secondary constraint has the singular goal of defining an optimal solution within the set of previously indistinguishable optimal solutions. It

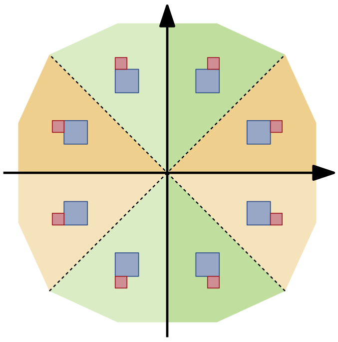


Figure 4.2: Case distinction on the separational constraints and m for the Demers cartogram

will therefore be multiplied with a suitably small constant before being added to the objective value of the linear program, which ensures that no new optimal solutions arise.

We want to measure the discrepancy in slope of the line through the two centers of two regions r_0, r_1 in the input and in the output. For this we introduce a new variable d_{r_0,r_1}, which measures the distance in the orthogonal direction of the separation constraint, as an approximate measure of the slope difference. The constraint uses the slope in the input which is precomputed as:

$$m = \frac{y(r_1) - y(r_0)}{x(r_1) - x(r_0)}$$

Depending on the separation of r_0 and r_1, different versions of the constraint are instantiated. These cases are also illustrated in Figure 4.2. Note that we always instantiate either the two constraints for horizontal separation or the two for vertical separation.

$$\begin{aligned}
&\text{[horizontal separation]}\\
&y(r_1) - y(r_0) < x(r_1) - x(r_0):\\
&d_{r_0,r_1} \geq y(r_1) - y(r_0) + m(x(r_1) - x(r_0)) && \text{for } y(r_1) \geq y(r_0)\\
&d_{r_0,r_1} \geq y(r_0) - y(r_1) + m(x(r_0) - x(r_1)) && \text{for } y(r_1) < y(r_0)
\end{aligned}$$

$$\begin{aligned}
&\text{[vertical separation]}\\
&y(r_1) - y(r_0) \geq x(r_1) - x(r_0):\\
&d_{r_0,r_1} \geq x(r_1) - x(r_0) + m(y(r_1) - y(r_0)) && \text{for } x(r_1) \geq x(r_0)\\
&d_{r_0,r_1} \geq x(r_0) - x(r_1) + m(y(r_0) - y(r_1)) && \text{for } x(r_1) < x(r_0)
\end{aligned}$$

We then minimize the sum over all of these variables, multiplied by a suitable small constant c_{nua} as previously mentioned. We additionally weigh this more for regions which are supposed to be adjacent. The idea is to lessen the importance of exact placement for regions which do not actually have a border in common.

$$\min \sum_{\{r_0,r_1\}\in\mathcal{R}} d_{r_0,r_1} \cdot c_{nua} \tag{4.7}$$

Note that this constraint penalizes the same difference in slope for regions which are placed farther apart more harshly than it does for regions which are closer together without being adjacent.

4.1.4 Displacement from Origin

If we want to keep a region r_0 at the position $p = (x(r_0), y(r_0))$ where they were placed in the input, we can add the following constraint.

$$o_{r_0} \geq |x_{r_0} - x(r_0)| \qquad \forall r_0 \in \mathcal{R} \tag{4.8}$$
$$o_{r_0} \geq |y_{r_0} - y(r_0)| \qquad \forall r_0 \in \mathcal{R} \tag{4.9}$$

Then minimize the variable o_r over all regions $r \in \mathcal{R}$. This variable is again multiplied by a suitably small constant c_{org}.

$$\min \sum_{r\in\mathcal{R}} o_r \cdot c_{org} \tag{4.10}$$

4.1.5 Extension to Multiple Cartograms and Stability Constraints

If we deal with time-series data we want to create multiple cartograms from the same input. For each of those cartograms we will create an LP. So far these LP's all have separate variables and constraints and can be solved independently. If we want to introduce the concept of stability, we want to create a connection through these constraints.

We try to achieve stability between two cartograms C_1, C_2 by minimizing the distance between the position of a polygon, in case of the Demers cartogram a square, of a region r_0 in C_1 and in C_2. This is done by instantiating the following constraint. The variable $x^i_{r_0}$ represent here the placement variable of r_0 in cartogram i.

$$t_{r_0} \geq |x^1_{r_0} - x^2_{r_0}| \qquad \forall r_0 \in \mathcal{R}$$
$$t_{r_0} \geq |y^1_{r_0} - y^2_{r_0}| \qquad \forall r_0 \in \mathcal{R}$$

This is as before summed up, multiplied by a weight c_{stab} and then minimized.

$$\min \sum_{r_0 \in \mathcal{R}} t_{r_0} \cdot c_{stab} \tag{4.11}$$

The cartograms are intended to be viewed as either a progression of cartograms, e.g., through time, or as a set of cartograms in which the user can switch from one cartogram to any other. These two views lead to two different approaches in trying to achieve stability. Let $[C_1, \ldots, C_k]$ be the list of cartograms we want to create. If we approach this as a series/progression of cartograms, we want to focus on stability from one point i in the progression to the next $i+1$. We call this the *successive stability model*. This means we actually instantiate the following constraint.

$$t^i_{r_0} \geq |x^i_{r_0} - x^{i+1}_{r_0}| \qquad \forall r_0 \in \mathcal{R}, \forall i \in [1, k-1] \tag{4.12}$$
$$t^i_{r_0} \geq |y^i_{r_0} - y^{i+1}_{r_0}| \qquad \forall r_0 \in \mathcal{R}, \forall i \in [1, k-1] \tag{4.13}$$

If, however, we view the cartograms as an unordered set where switching can occur between all pairs of cartograms, we need to generalize this to the following constraint.

$$t^{i,j}_{r_0} \geq |x^i_{r_0} - x^j_{r_0}| \qquad \forall r_0 \in \mathcal{R}, \forall i, j \in [1, k-1], i \neq j \tag{4.14}$$
$$t^{i,j}_{r_0} \geq |y^i_{r_0} - y^j_{r_0}| \qquad \forall r_0 \in \mathcal{R}, \forall i, j \in [1, k-1], i \neq j \tag{4.15}$$

The second approach, which we call the *complete stability model*, might be viewed as a more holistic approach that can account for stability across the whole set. The obvious downside to this is that, the number of created constraints in the second case is in $\mathcal{O}(n \cdot k^2)$, while the first is in $\mathcal{O}(n \cdot k)$.

Another downside to both of these approaches, in contrast to just creating a single cartogram is that the total size of the LP increases with a factor of k. To counteract this we can employ an iterative approach to creating stable cartograms.

Iterative Approach

In this approach we do not want to connect the LP's into one big LP, but rather keep them separate and use the result of one LP as a basis to guarantee stability to the next. For this we create the LP for cartogram C_1 as described in sections 4.1.1 to 4.1.4 and solve it. We then fix the positions of all regions in C_1. Then we employ the minimization of the distance to the origin, not by using the position in the input but the already fixed position in the last solved LP for the next. In general we create the following constraint.

$$t^i_{r_0} \geq |X^i_{r_0} - x^{i+1}_{r_0}| \qquad \forall r_0 \in \mathcal{R}, \forall i \in [1, k-1] \tag{4.16}$$
$$t^i_{r_0} \geq |Y^i_{r_0} - y^{i+1}_{r_0}| \qquad \forall r_0 \in \mathcal{R}, \forall i \in [1, k-1] \tag{4.17}$$

Note that $X^i_{r_0}, Y^i_{r_0}$ are constant in contrast to the variables x^{i+1}, y^{i+1}. This means that the placement in cartogram $i+1$ cannot influence the placement in cartogram i. This approach will be called the *iterative stability model*.

4.2 LP-Model for iso-hexagon cartograms

In this section we describe the changes we need to make to the formalization in order to create iso-hexagon cartograms instead of Demers cartograms. In particular we will introduce new variables which account for the additional drawing directions and adapt constraints to use these new variables.

4.2.1 Position of Regions

Additionally to the already defined variable x_{r_0}, y_{r_0} we define three variables $z_0(r), z_1(r), z_2(r)$ which redundantly identify the placement of a region r in the plane. The corresponding directions are depicted in Figure 4.3a.

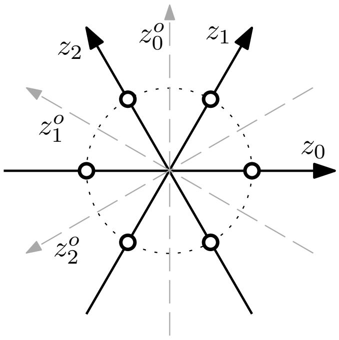
$$z_0(r) = x_r \tag{4.18}$$
$$z_1(r) = \cos\left(\frac{\pi}{3}\right) \cdot x_r + \sin\left(\frac{\pi}{3}\right) \cdot y_r \tag{4.19}$$
$$z_2(r) = \cos\left(2 \cdot \frac{\pi}{3}\right) \cdot x_r + \sin\left(2 \cdot \frac{\pi}{3}\right) \cdot y_r \tag{4.20}$$

4.2.2 Disjointedness and Separation Constraints

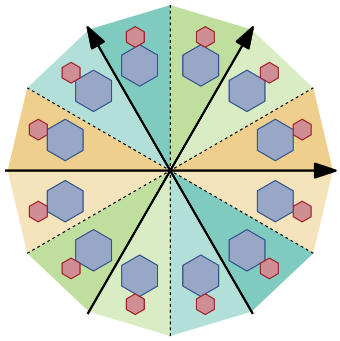
To enforce this distance on the regions we add the following constraints into the LP. Note that the structure of these constraints is similar to the ones used in the Demers cartogram LP, just adapted to the new positional variables z_0, z_1, z_2.

$$z_0(r_1) - z_0(r_0) \geq D^{min}_{01} + gap_{r_0,r_1} \qquad \forall (r_0, r_1) \in S_0 \tag{4.21}$$
$$z_1(r_1) - z_1(r_0) \geq D^{min}_{01} + gap_{r_0,r_1} \qquad \forall (r_0, r_1) \in S_1 \tag{4.22}$$
$$z_2(r_1) - z_2(r_0) \geq D^{min}_{01} + gap_{r_0,r_1} \qquad \forall (r_0, r_1) \in S_2 \tag{4.23}$$

In order to keep adjacent regions adjacent in the output, we want to add the following constraints. The variable $s_0^{r_0,r_1}, s_1^{r_0,r_1}, s_2^{r_0,r_1}$ measure the distance between r_0 and r_1. For



(a)



(b)

Figure 4.3: (a) Orthogonal directions of the hexagonal case are not yet encoded in the system. (b) Case distinction on the separational constraints and m for the hexagonal case.

ease of notation we will write $\delta_i(r_0, r_1)$ for $z_i(r_0) - z_i(r_1)$.

$$s_0^{r_0,r_1} \geq \max(\delta_0(r_0,r_1) - D_{01}^{min}, \delta_0(r_0,r_1) - D_{01}^{min}) \qquad \forall\{r_0,r_1\} \in \mathcal{T} \tag{4.24}$$

$$s_1^{r_0,r_1} \geq \max(\delta_1(r_0,r_1) - D_{01}^{min}, \delta_1(r_0,r_1) - D_{01}^{min}) \qquad \forall\{r_0,r_1\} \in \mathcal{T} \tag{4.25}$$

$$s_2^{r_0,r_1} \geq \max(\delta_2(r_0,r_1) - D_{01}^{min}, \delta_2(r_0,r_1) - D_{01}^{min}) \qquad \forall\{r_0,r_1\} \in \mathcal{T} \tag{4.26}$$

$$s_0^{r_0,r_1}, s_1^{r_0,r_1}, s_2^{r_0,r_1} \geq 0 \qquad \forall\{r_0,r_1\} \in \mathcal{T} \tag{4.27}$$

Now we can minimize these distances.

$$\min \sum_{\{r_0,r_1\}\in\mathcal{T}} s_0^{r_0,r_1} + s_1^{r_0,r_1} + s_2^{r_0,r_1} \tag{4.28}$$

4.2.3 Slope

To generalize this idea to the hexagonal case, we need some further information about the placement of the regions, namely the coordinates in the orthogonal directions to the three defined coordinates z_0, z_1, z_2. Observe in Figure 4.3a that these are not yet encoded in our system. We define the orthogonal directions z_0^o, z_1^o, z_2^o as follows.

$$z_0^o(r) = y(r) \tag{4.29}$$

$$z_1^o(r) = -\sin\left(\frac{\pi}{3}\right) \cdot x(r) + \cos\left(\frac{\pi}{3}\right) \cdot y(r) \tag{4.30}$$

$$z_2^o(r) = -\sin\left(2 \cdot \frac{\pi}{3}\right) \cdot x(r) + \cos\left(2 \cdot \frac{\pi}{3}\right) \cdot y(r) \tag{4.31}$$

Next we want to formulate the case distinction for the hexagonal case. We will differentiate not just $m \geq 0$ vs. $m < 0$ but instead will define 6 equally sized ranges in which the slope can fall, each spanning $[\alpha, \alpha + \frac{\pi}{3}[$, where $\alpha \in \left\{0, \frac{\pi}{3}, \frac{2\pi}{3}, \pi, \frac{4\pi}{3}, \frac{5\pi}{3}\right\}$. This defines 6 different cases which we will call A_i, B_i, C_i for $i \in \{1, 2\}$, illustrated in Figure 4.3b.

The slope variable m is going to be adjusted into the following three variables.

$$
\begin{aligned}
m_0 &= m \\
m_1 &= \tan(\theta - \frac{\pi}{3}) \\
m_2 &= \tan(\theta - \frac{2\pi}{3})
\end{aligned}
$$

These definitions make use of the angle $\theta = \text{atan2}(1/m, m)$. The $\text{atan2}(x, y)$ function, which calculates the angle between the x-axis and a line through $(0, 0)$ and (x, y), is defined as:

$$
\text{atan2}(x, y) = \begin{cases}
\arctan\left(\frac{x}{y}\right) & \text{if } x > 0 \\
\arctan\left(\frac{x}{y}\right) + \pi & \text{if } x < 0 \land y \geq 0 \\
\arctan\left(\frac{x}{y}\right) - \pi & \text{if } x < 0 \land y < 0 \\
\frac{\pi}{2} & \text{if } x = 0 \land y > 0 \\
-\frac{\pi}{2} & \text{if } x = 0 \land y < 0 \\
\text{undefined} & \text{if } x = 0 \land y = 0
\end{cases}
$$

The two variables $z_0(r)$ and $z_0^o(r)$ encode exactly the x and y coordinates of the center point p^r of the region r. With $z_1(r), z_1^o(r)$ and $z_2(r), z_2^o(r)$ we encode these coordinates in relation to a coordinate system which is rotated by 60° and 120° respectively.

$$
\forall i \in \{0, 1, 2\} : \begin{cases}
z_i(p^r) = x(p_i^r) \\
z_i^o(p^r) = y(p_i^r)
\end{cases}
$$

The newly defined variables m_0, m_1 and m_2 encode the slopes of the lines through the origin and $p_0^r = p_r$, p_1^r and p_2^r respectively, which are just rotations around the origin by 0°, 60° and 120° of the line through the origin and p^r.

With these newly defined variables, we can formulate the generalized constraint for the hexagonal case, which is again differentiated in relation to the membership of r_0 and r_1 in S_0, S_1 and S_2. If (r_0, r_1) is contained in multiple separation constraints, we instantiate the constraint for S_k s.t. $\forall i \in \{0, 1, 2\} : |z_k(r_0) - z_k(r_1)| \geq |z_i(r_0) - z_i(r_1)|$, i.e., we choose the dimension in which r_0 and r_1 have the greatest separation.

$$
\begin{aligned}
&(r_0, r_1) \in S_0 \lor (r_1, r_0) \in S_0 : \\
&d_{r_0,r_1} \geq z_0^o(r_1) - z_0^o(r_0) + m_0(z_0(r_1) - z_0(r_0)) \qquad \text{for } z_0^o(r_1) \geq z_0^o(r_0) \qquad (4.32) \\
&d_{r_0,r_1} \geq z_0^o(r_0) - z_0^o(r_1) + m_0(z_0(r_0) - z_0(r_1)) \qquad \text{for } z_0^o(r_1) < z_0^o(r_0) \qquad (4.33)
\end{aligned}
$$

$$(r_0, r_1) \in S_1 \vee (r_1, r_0) \in S_1 :$$

$$d_{r_0,r_1} \geq z_1^o(r_1) - z_1^o(r_0) + m_1(z_1(r_1) - z_1(r_0)) \qquad \text{for } z_1^o(r_1) \geq z_1^o(r_0) \tag{4.34}$$

$$d_{r_0,r_1} \geq z_1^o(r_0) - z_1^o(r_1) + m_1(z_1(r_0) - z_1(r_1)) \qquad \text{for } z_1^o(r_1) < z_1^o(r_0) \tag{4.35}$$

$$(r_0, r_1) \in S_2 \vee (r_1, r_0) \in S_2 :$$

$$d_{r_0,r_1} \geq z_2^o(r_1) - z_2^o(r_0) + m_2(z_2(r_1) - z_2(r_0)) \qquad \text{for } z_2^o(r_1) \geq z_2^o(r_0) \tag{4.36}$$

$$d_{r_0,r_1} \geq z_2^o(r_0) - z_2^o(r_1) + m_2(z_2(r_0) - z_2(r_1)) \qquad \text{for } z_2^o(r_1) < z_2^o(r_0) \tag{4.37}$$

Constraints (4.32)-(4.37) can be formulated as one generalized constraint:

$$(r_0, r_1) \in S_i \vee (r_1, r_0) \in S_i :$$

$$d_{r_0,r_1} \geq z_i^o(r_1) - z_i^o(r_0) + m_i(z_i(r_1) - z_i(r_0)) \qquad \text{for } z_i^o(r_1) \geq z_i^o(r_0) \tag{4.38}$$

$$d_{r_0,r_1} \geq z_i^o(r_0) - z_i^o(r_1) + m_i(z_i(r_0) - z_i(r_1)) \qquad \text{for } z_i^o(r_1) < z_i^o(r_0) \tag{4.39}$$

4.2.4 Other Differences

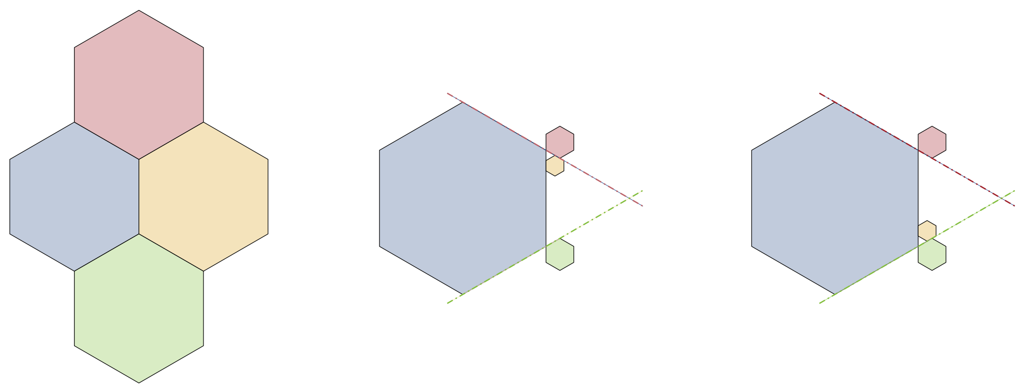
We further stress that we do not define an equivalent to the strong case for the iso-hexagon cartograms. All other constraints we have described in Section 4.1 can be instantiated as before.

CHAPTER 5

Leader Lines

The necessity of separation constraints and the limitations of contact representations can lead to situations in which adjacencies between regions are lost. As a cartogram tries to resemble the topology of the input, lost adjacencies are undesirable. The LP model presented in Chapter 4 tries to minimize the distance between regions, which should be adjacent, in an attempt to minimize the number of lost adjacencies.

However it is possible to define the adjacency graph $G = (V, E)$ and assign weights to regions, such that not all adjacencies can be realized in a Demers cartogram or a iso-hexagon cartogram. These cases are depicted in Figures 3.3, 3.4 and 5.1. A way to cope with this is to visualize lost adjacencies through connecting curves between the polygons of regions, which are supposed to be adjacent.



(a) (b) (c)

Figure 5.1: Adjacent regions in Cartogram (a) lose at least one adjacency in a new Cartogram, where their weights have changed (b-c).

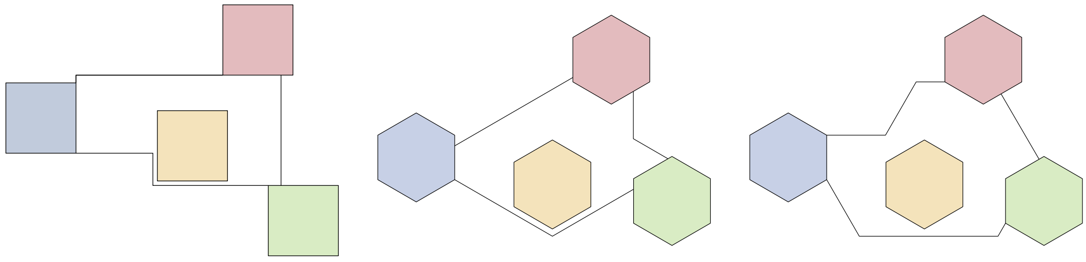


Figure 5.2: Leader lines in a Demers cartogram (a), and leader lines which run orthogonal (b) and parallel (c) to the directions z_0, z_1, z_2 in a iso-hexagon cartogram.

5.1 Definition

We will call these curves *leader lines*. In order to keep the visual style of the cartogram, we require every section of a leader line to be orthogonal to one of the available drawing dimensions. Examples are presented in Figures 5.2a and 5.2b. Note that in a Demers cartogram, lines running parallel to the drawing directions are equivalent to lines running orthogonal to the drawing directions. However since no drawing direction in a iso-hexagon cartogram is orthogonal to another, we want to keep the leader lines orthogonal to the drawing directions, which keeps them parallel to the edges of the hexagons and lowers visual disturbance. Figure 5.2c is an example in which the leader lines are kept parallel to the drawing directions.

5.2 Leader Lines in a Demers cartogram

Nickel et al. [NSM$^+$19] showed that lost adjacencies between regions r_1 and r_2 in a Demers cartogram, created by the linear program discussed in Section 4.1, can be visualized with an orthogonal line l of bounded length. They prove an upper bound of the length of this leader line, which is equal to the distance between the closest points of r_1 and r_2 we want to connect, measured in L_1-metric. In the next section we want to go into the details of the proof, since we will extend some of the concepts to the iso-hexagon cartograms.

5.2.1 Existence of a Minimal Length Leader Line

First they assume that $(r_1, r_2) \in S_x$, i.e., r_2 ought to be placed right of r_1. They also assume the existence of a hypothetical cartogram $\mathcal{A}$, in which the adjacency between r_1 and r_2 can be realized, and they assume r_1 and r_2 to be minimal, in the sense that there can not be a third region r_3, s.t., $(r_1, r_3), (r_3, r_2) \in S_x$. Next they differentiate the relative placement of r_1 and r_2. If there exists a horizontal line that intersects both regions, this line cannot be interrupted by another square due to the minimality of r_1 and r_2 and the line has the required minimal length. If r_2 is placed completely above r_1, they define an area $S = [x_1, x_2] \times [y_1, y_2]$ which spans between the upper right corner of

r_1 and the lower left corner of r_2. The leader l must be completely contained in S. Note that the lower right corner of S is the extreme point at which a square, that was above r_1 and left of r_2, can be drawn. Conversely the upper left corner is the extreme point for square below r_2 and right of r_1.

Then they identify the set of all regions L, s.t., $r_l \in L \implies (r_1, r_l) \in S_y \wedge (r_l, r_2) \in S_x$, i.e., all regions which were placed above r_1 and left of r_2. Note two things. First, this set is a subset of the set L^{max} of all regions, which have their center left of the line connecting the centers of r_1 and r_2 in $\mathcal{A}$ and second, no square in $L^{max} \backslash L$ can interfere with the leader line. The area L is subtracted from S resulting in a (non-empty) region $S' = S \backslash L$. Finally the leader is obtained by tracing the upper boundary of S'. This results in a line that starts out vertically and turns right into a horizontal segment, if a region $r \in L$ restricts the area at this height, until it reaches the corner of r then turning left again into a vertical segment, forming a concave bend in the line. The region r is called *responsible* for the bend.

The existence of l is argued by assuming that there exists a region obstructing this line and deriving a contradiction. They assume r is an obstacle of this line. Note that r can not be in L, since it would have otherwise simply been responsible for a concave bend in l and the l would avoid the obstacle by definition of S'. The square r must intersect l at some segment. Since l is orthogonal, this is either a vertical or a horizontal segment. Assume it intersects l at a horizontal segment and let r' be the square responsible for the concave bend in l. If no such bend exists then $r' = r_2$. The intersection of r and l at a horizontal segment directly implies that r and r' can not be separated by a horizontal line and $(r', r) \in S_x$ would make this intersection impossible. Therefore we know that $(r, r') \in S_x$. Since $r' \in L$ we know that $(r', r_2) \in S_x$ and since separation constraints are transitive, we know that $(r, r_2) \in S_x$. In order to be able to interfere with the line, r can not be placed below or to the left of r_1. If $(r_1, r) \in S_y$, then r would be in L, which, as already argued, is impossible. If $(r_1, r) \in S_x$, then r_1 and r_2 are not minimal. Therefore r can not exist.

5.2.2 Differences in the Iso-Hexagon Case

Even though we will use some of the ideas presented by Nickel et al. [NSM$^+$19], it is important to understand that this proof does not simply translate to the iso-hexagon cartograms. In this section we want to point to the differences, which require attention. First, if in a Demers cartogram two adjacent regions r_1, r_2 are placed next to each other, e.g., $(r_1, r_2) \in S_x$, r_1, r_2 are minimal and they have a horizontal line stabbing through both regions, then, as argued above, this line cannot be possibly intersected by third region, regardless of the horizontal distance between r_1 and r_2. However, in a iso-hexagon cartogram, this is not guaranteed. If $(r_1, r_2) \in S_0$ and a third region r_3 exists such that $(r_1, r_3) \in S_1 \wedge (r_2, r_3) \in S_2$, then increasing the distance between r_1 and r_2 can lead to a placement of r_3, which is consistent with the separation constraints, but nevertheless completely blocks any horizontal line between r_1 and r_2. This is illustrated in Figure 5.3a. Even an adjustment to this region as shown in Figure 5.3b does not

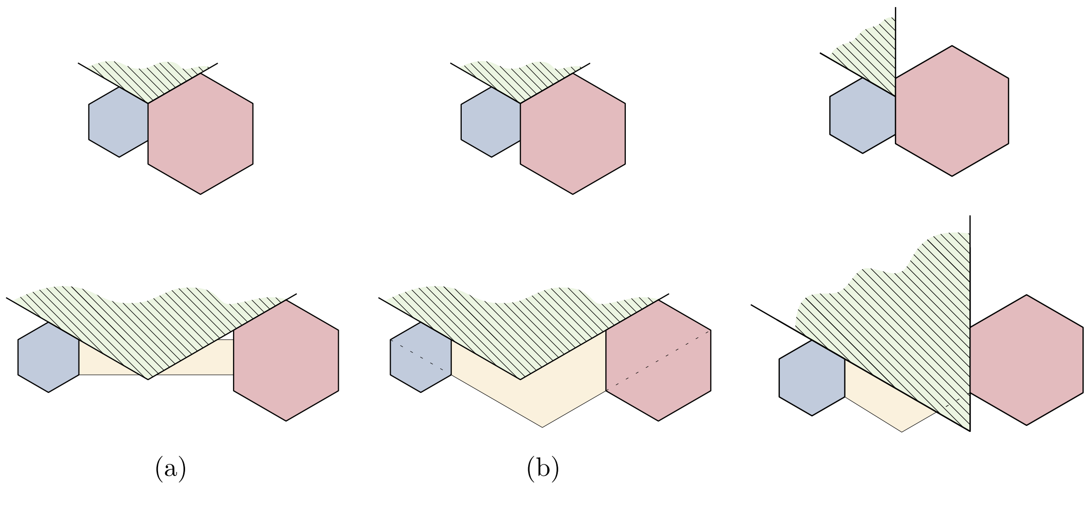


Figure 5.3: (a) No horizontal line exists between the two regions. Even adjusting the area, s.t., it has edge-parallel boundaries (b) does not guarantee that the area contains a valid leader line, since the area does not actually model the extreme points of placement for obstacles, which is shown in (c).

trivially solve this problem, since in contrast to the Demers cartogram, an obstacle in a iso-hexagon cartogram can completely block the edge of r_0 which had contact with r_1, as displayed in Figure 5.3c.

In the next section we proof an upper bound of a leader line in a iso-hexagon cartogram, by following similar steps as presented above and amending the argument where necessary.

5.3 Leader Lines in a iso-hexagon cartogram

Before we start analyzing the leader lines in a iso-hexagon cartogram, we want to define some additional notation, which enables us to talk about the hexagons, representing the regions.

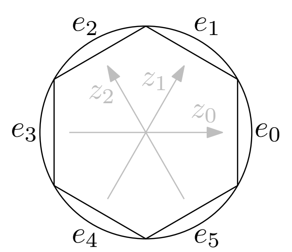


Figure 5.4

Since all hexagons representing regions are rotated the same way, we can define a lexicographical order on the edges. The rightmost edge of a region r_0 will be called $e_0(r_0)$ and we number the edges in counter-clockwise order, as depicted in Figure 5.4. We will always consider the index of the edge to be interpreted as being modulo 6. An edge e_i is always perpendicular to the direction z_j, with $j = i \mod 3$. We will further use in- and decrementation on the separation constraints in this chapter. For this we define $(r_1, r_2) \in S_{2+1} \implies (r_2, r_1) \in S_0$ and also $(r_1, r_2) \in S_{0-1} \implies (r_2, r_1) \in S_2$, i.e., we consider the separation constraints to be interpreted as being modulo 3.

We assume, that there exists a hypothetical Cartogram $\mathcal{A}$ without size constraints in which all adjacencies and all separation constraints are kept. The assumption of existence of such a cartogram is sufficient and we do not need to know the specific placement of the regions in $\mathcal{A}$.

In order to talk about the distances of two regions we define the following values.

$$\delta_i(r_0, r_1) = z_i(r_0) - z_i(r_1)$$

$$\delta_i^o(r_0, r_1) = z_i^o(r_0) - z_i^o(r_1)$$

$$k_i(r_0, r_1) = \delta_i(r_0, r_1) - \frac{(s_i(r_0) + s_i(r_1))}{2}$$

The value $k(r_0, r_1)$ is the distance of r_0 and r_1 in the dimension of biggest separation, measured from the border of the polygons.

Let $c(r)$ be the center of region r. Let $s_i(r)$ be the line through $c(r)$, parallel to z_i^o. Note that $s_i(r)$ crosses through two opposing corners and splits r in half. Let $l_i(r)$ be the line through both endpoints of $e_i(r)$. Let l, l' be two lines. Then we define the relations $>_i$ and $\geq_i$ as follows:

$$l \geq_i l' \iff \forall p \in l, p' \in l' : z_i(p) \geq z_i(p')$$

$$l >_i l' \iff \forall p \in l, p' \in l' : z_i(p) > z_i(p')$$

Intuitively we want to be able to define an ordering on parallel lines in a given dimension.

And lastly we want to proof a property of adjacent regions in the hypothetical cartogram $\mathcal{A}$. Assume that $(r_0, r_1) \in S_1$ and that $\{r_0, r_1\} \in \mathcal{T}$, i.e., they are adjacent in $\mathcal{A}$. From this we can follow this lemma.

Lemma 5.3.1. *If two regions r_0 and r_1 are adjacent and $(r_0, r_1) \in S_i$, then $e_{i-1}(r_0) \geq_{i-1} e_{i+2}(r_1)$ and $e_{i+1}(r_0) \geq_{i+1} e_{i-2}(r_1)$ in $\mathcal{A}$.*

Proof. Assume, w.l.o.g., that $(r_0, r_1) \in S_1$. Assume further $e_0(r_0) <_0 e_3(r_1)$. Then $\forall p \in e_0(r_0), q \in e_3(r_1) : z_0(p) < z_0(q)$. Note that all points on $e_0(r_0)$ have a maximal z_0-coordinate in the set of all points in r_0 and all points on $e_3(r_1)$ have a minimal z_0-coordinate in the set of all points in r_1. We now draw a line l orthogonal to z_0 with the z_0-coordinate $\frac{q+p}{2}$. Note that l is parallel to $e_0(r_0)$ and $e_3(r_1)$ and that $\forall t \in l : z_0(p) < z_0(t) < z_0(q)$. The line l is therefore a separating line between r_0 and r_1, which makes them not adjacent. This is a contradiction to the assumption that $\mathcal{A}$ keeps all adjacencies. The same argument holds for $e_2(r_0) \geq_2 e_5(r_1)$ and all other directions are equal up to rotation and symmetry. □

And finally we ask the reader to note that separation constraint membership can be determined by the relative center placement of two hexagons. Dependent on the placement of the center of one hexagon r' relative to another r, we can determine their separation

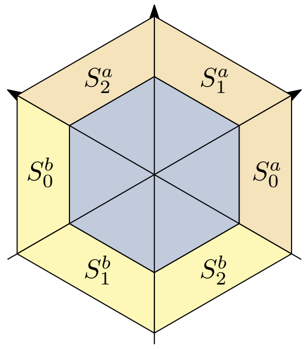


Figure 5.5: Sectors of a hexagon of region r defined by lines through opposing corners. If the center of another hexagon of region r' lies in Sector S_i^a we have $(r, r') \in S_i$, if it lies in a Sector S_i^b we have $(r', r) \in S_i$.

constraint. The sectors, as shown in Figure 5.5, of a hexagon of region r are defined by lines through opposing corners. If the center of another hexagon of region r' lies in sector S_i^a we have $(r, r') \in S_i$, if it lies in sector S_i^b we have $(r', r) \in S_i$.

5.3.1 Proof of a Leader Line

We want to define an area S between the two regions r_0, r_1 which are supposed to be connected by a leader line l, s.t., l is contained in S. In a Demers cartogram this was done by defining an area that captures the extreme placement options of obstacles on either side of this line. Before we can define this area, we want to look at what regions can actually pose an obstacle to a leader line between r_0 and r_1. We group these obstacle in two groups L and R. L contains all regions with their centers placed on one side (left) of $l_{cent} = \overline{c(r_0)c(r_1)}$ and R contains all regions on the other side (right) of l_{cent}. Note that all regions $r_l \in L$ must have one of four separation constraints with r_0 (and r_1), namely

$$(r_0, r_l) \in S_0 \vee (r_0, r_l) \in S_1 \vee (r_0, r_l) \in S_2 \vee (r_l, r_0) \in S_0$$

Assume $(r_0, r_1) \in S_0$. Then we want a leader line to be contained in the area between the two regions, i.e., two one side of $l_0(r_0)$ and to the other side of $l_3(r_1)$. We call this area A. Because of this we can exclude all regions r_l with $(r_l, r_0) \in S_0$ from L since they can not possibly intersect A.

$$L' = L \backslash \{r_l | (r_l, r_0) \in S_0\}$$

R' is defined accordingly. Next we state the following lemma. Less technically, this lemma states that two regions from L' and R' respectively, can not cross inside A.

Lemma 5.3.2. *Let $r_l \in L'$ and $r_r \in R'$. If $\{r_l, r_r\} \in \mathcal{T}$, then either $(r_0, r_l) \in S_2 \wedge (r_r, r_0) \in S_1$ or $(r_1, r_l) \in S_1 \wedge (r_r, r_1) \in S_2$.*

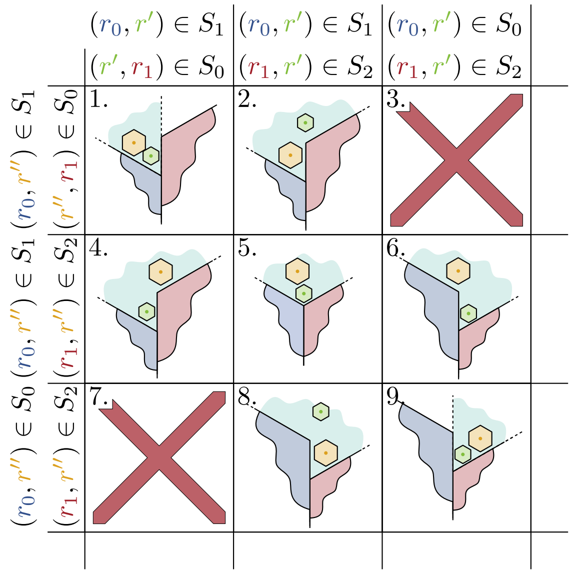


Figure 5.6: All nine different combinations of memberships of two regions $r', r'' \in L^{min}$.

Proof. Obvious because of the planar embedding of the adjacency graph where every vertex representing a region is placed exactly where the region is placed in $\mathcal{A}$. □

We define the two sets $L^{min} = L' \backslash \{r_l | (r_0, r_l) \in S_2 \vee (r_1, r_l) \in S_1\}$ and $R^{min} = R' \backslash \{r_r | (r_r, r_0) \in S_1 \vee (r_r, r_1) \in S_2\}$. From the previous lemma we know that the regions in L^{min} and R^{min} are pairwise not adjacent.

Placement of Obstacles on the Same Side

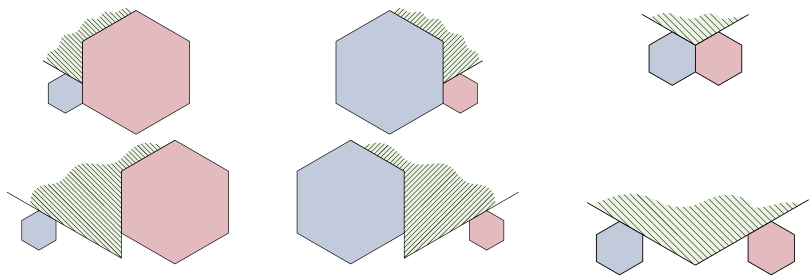
We will now look at the possible extreme placements of regions both L^{min} and R^{min}. We will first analyze how the placement of an obstacle affects the possible placement of other obstacles in the same set. For this we present Figure 5.6. There we can clearly see that a configuration in which the first region r' is to the left of r_0 and the second r'' is to the right of r_1, i.e., $(r_0, r'), (r'', r_1) \in S_0$, is impossible. This can be proven by using the fact that for this to happen, we need both r_0 and r_1 to extend above the other one, which is a contradiction. That means that, if we draw the possible positions of obstacle regions in L^{max}, we can encounter either the possible placement in Figure 5.7a or 5.7b but never both at the same time, since the vertical line on the left of the placement area is enforced by $(r_0, r') \in S_0$ and the right one by (r'', r_1). Note that the placement area in Figure 5.7c is strictly contained in the other two. The same area can be defined for the mirrored version of R^{max}. We will treat this areas as being completely filled in an extreme case.

Placement of Obstacles on Different Sides

If we consider all possible combinations of placement areas for L^{min} and R^{min}, we arrive at the four possibilities displayed in figure 5.8. Note that the union of both areas in Figure 5.8a is completely contained in the union of both areas in Figure 5.8c. The same holds for Figures 5.8b and 5.8d. Figures 5.8c and 5.8d give us a parallelogram in which we want to draw the leader line l. Let A be that parallelogram.

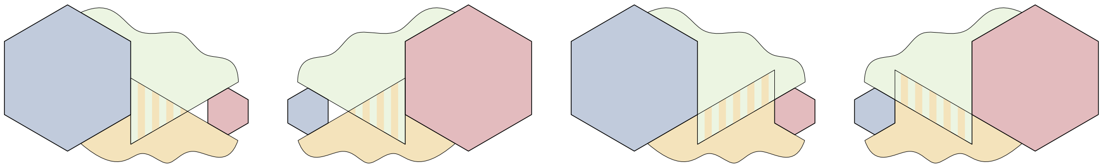
In the first case (Figure 5.8c) we adapt this area by excluding all points p that lie, in a sense, above the lowest corner of a region $r_l \in L^{min}$. Let q be the lowest corner of r_l, i.e., the crossing point of $e_4(r_l)$ and $e_5(r_l)$. Then excluding all points p above q, i.e., $z_1(p) > z_1(q) \wedge z_2(p) > z_2(q)$, results in an area $A' \subseteq A$. The leader line is obtained by tracing the upper boundary of A'. In the second case (Figure 5.8d) we exclude the points lying below the upper corners of hexagons $r_r \in R'$ in a similar manner, we end up with our final region A''. The leader line is obtained by tracing the lower boundary of A''. Figure 5.9 is an example of a leader line construction for A'. If the leader would follow the dashed line, it would have half the length of the boundary of A', which is the upper bound of a leader line constructed in this way. Since A' can be filled with obstacles up to arbitrarily small ε-sized gaps, this bound is tight, even though it will never be reached.

The existence of this leader line inside the area A follows directly from the fact that no two regions $r_l \in L^{min}, r_r \in R^{min}$ can be adjacent as stated before. The existence of a leader line as constructed above, however is not as clear. In order to proof this we first want to establish a lemma about the possible separation constraints between regions in L^{min} and R^{min}. This is done in the next section.



(a) (b) (c)

Figure 5.7: Placement areas for regions in L^{min}. In an extreme case, the union of all regions in L^{min} could fill the complete placement area, up to ε-sized holes which can be arbitrarily small. Note that the area of (c) is strictly included in (a) and (b).



(a) (b) (c) (d)

Figure 5.8: All four possible combinations of placement areas for regions in L^{min} and R^{min}. The overlap of the areas is marked with the shaded areas. The union of both areas in (a) and (b) is included in (c) and (d) respectively.

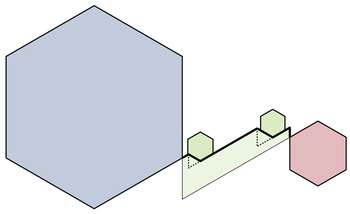


Figure 5.9: Leader line example. The dashed lines indicate the upper bound.

Separation constraints of Obstacles on Different Sides

We want to answer the question if, for every pair of hexagons $r_l \in L^{min}, r_r \in R^{min}$, we can always find a separating line between them that is either parallel to $l_1(r_0)$ or $l_5(r_0)$. To show this it is, in turn, sufficient to show the following

$$\forall r_l \in L^{max}, r_r \in R^{max} : (r_r, r_l) \in S_1 \cup S_2$$

Here we ask the reader to note that separation constraints are equivalent to relative center placement and that all hexagons which are placed in two lower sectors of a hexagon (see Figure 5.5) have a separating line parallel to $l_1(r_0)$ or $l_5(r_0)$ with that hexagon.

To simplify the argumentation below we want to establish two lemmas, which capture the two core concepts used in the proof below. Lemma 5.3.3 states that for two regions r, r' with a separation constraint $(r, r') \in S_i$, we can always find a separating line between $s_i(r')$ and $s_i(r)$. Lemma 5.3.4 states that in certain constellations of three regions r_0, r_1, r_l, e.g., the first and third column of Figure 5.11, we can also find a separating line between a different set of lines through their centers. Figure 5.10 is an example of such a constellation and the two lines for which the lemma states, they have a separating line, are highlighted.

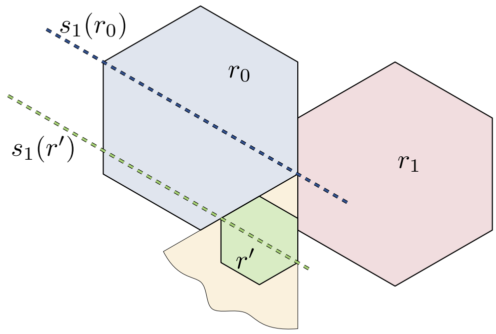


Figure 5.10

Lemma 5.3.3. *Let r and r' be two regions, s.t., $(r, r') \in S_i$. Then $s_i(r') >_i s_i(r)$.*

Proof. This first lemma is straight forward. Since all regions have non-zero size we know $s_i(r') >_i e_{i+3}$ and $e_i(r) >_i s_i(r)$. The separation constraint $(r, r') \in S_i$ enforces $e_{i+3} >_i e_i(r)$. From this it clearly follows that $s_i(r') >_i e_{i+3} >_i e_i(r) >_i s_i(r)$. □

Lemma 5.3.4. *Let r_0, r_1 and r' be regions, s.t., $(r_0, r_1) \in S_0$ and $\{r_0, r_1\} \in \mathcal{T}$.*

1. *If $(r', r_1) \in S_0$*
   a) *If $(r', r_0) \in S_2$, then $s_1(r_0) >_1 s_1(r')$.*
   b) *If $(r_0, r') \in S_1$, then $s_2(r') >_2 s_2(r_0)$.*
2. *If $(r_0, r') \in S_0$*
   a) *If $(r', r_1) \in S_1$, then $s_2(r_0) >_2 s_2(r')$.*
   b) *If $(r_1, r') \in S_2$, then $s_1(r') >_1 s_1(r_0)$.*

Proof. We will give the proof for case 1.a). All other cases are equal up to symmetry. We define the area A_p as all points left of $e_3(r_1)$ and below $e_5(r_0)$. This area is marked in Figure 5.10.

$$\left.\begin{array}{l} (r', r_0) \in S_2 \implies e_5(r_0) >_2 s_2(r') \\ (r', r_1) \in S_0 \implies e_3(r_0) >_0 s_0(r') \end{array}\right\} \implies c(r') \in A_p$$

By definition of A_p we know that every line l_p parallel to z_1^o, which runs through a point in A_p lies below $s_1(r_0)$. Since $c(r') \in A_p$ and $c(r') \in s_1(r')$ we know that $s_1(r_0) >_1 s_1(r')$. □

We present Figure 5.11, in which all possible combinations of separation constraints between r_l—in green and at the top—and r_r—in orange and at the bottom—are shown.

Recall the assumption of the existence of $\mathcal{A}$. In this hypothetical cartogram, r_0 and r_1 are placed adjacent to each other and all separation constraints,In particular the separation constraint between r_l and r_r, are kept. We will show that in $\mathcal{A}$ (and therefore in every other cartogram) this separation constraint can only ever be S_1 or S_2 and never S_0.

Now we state the following Theorem, which restricts the separation constraints of obstacles on different sides of l_{sep}.

Theorem 5.3.5. *Let r_0, r_1, r_l, r_r be four regions, s.t. $\{r_0, r_1\} \in \mathcal{T}$, $(r_0, r_1) \in S_i$ and $r_l \in L^{min}, r_r \in R^{min}$. Then $(r_l, r_r), (r_r, r_l) \notin S_i$ in $\mathcal{A}$.*

Proof. We will give the proof for $(r_0, r_1) \in S_0$. All other cases are equal up to symmetry. First note that from $s_1(r_l) >_1 s_1(r_r)$ it directly follows that $(r_l, r_r) \notin S_i$ and from $s_2(r_l) >_2 s_2(r_r)$ it directly follows that $(r_r, r_l) \notin S_i$. Next we know from Lemma 5.3.1 that $e_1(r_0) >_1 e_4(r_1)$ and $e_2(r_1) >_2 e_5(r_0)$.

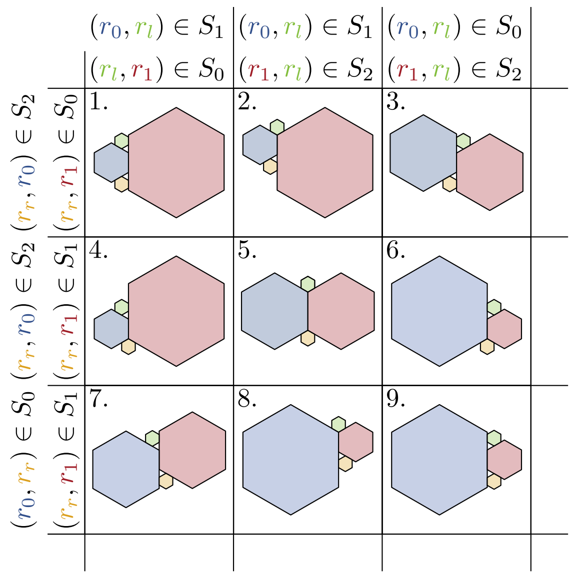


Figure 5.11: Nine different possible configurations of the two regions $r_l \in L^{min}$ and $r_r \in R^{min}$. The two regions are not necessarily adjacent to r_0 or r_1.

We will now show for every possible configuration in Figure 5.11 that the following holds:

$$(I) : s_1(r_l) >_1 s_1(r_r)$$
$$(II) : s_2(r_l) >_2 s_2(r_r)$$

Configuration 1 (and symmetrically configuration 9):

(I): By Lemma 5.3.3 we know that $s_1(r_l) >_1 s_1(r_0)$. By Lemma 5.3.4 we know that $s_1(r_0) >_1 s_1(r_r)$.

(II): By Lemma 5.3.4 we know that $s_2(r_l) >_2 s_2(r_0)$. By Lemma 5.3.3 we know that $s_2(r_0) >_2 s_2(r_r)$.

Configuration 2 (and symmetrically configurations 4, 6, 8):

(I): By Lemma 5.3.3 we know that $s_1(r_l) >_1 s_1(r_0)$. By Lemma 5.3.4 we know that $s_1(r_0) >_1 s_1(r_r)$. This is the same as for 1.I.

(II): By separation constraint we know that $s_2(r_l) >_2 e_2(r_1)$ and $e_5(r_0) >_2 s_2(r_r)$. By Lemma 5.3.1 we have $e_2(r_1) >_2 e_5(r_0)$.

Configuration 3 (and symmetrically configuration 7):

(I): By Lemma 5.3.4 we know $s_1(r_l) >_1 s_1(r_0)$ and $s_1(r_1) >_1 s_1(r_r)$. As a direct consequence of Lemma 5.3.1 we know $s_1(r_0) >_1 s_1(r_1)$

(II): By separation constraint we know that $s_2(r_l) >_2 e_2(r_1)$ and $e_5(r_0) >_2 s_2(r_r)$. By Lemma 5.3.1 we have $e_2(r_1) >_2 e_5(r_0)$. This is the same as for 2.II.

Configuration 5:

(I): By separation constraint we know that $s_1(r_l) >_1 e_1(r_1)$ and $e_4(r_0) >_2 s_1(r_r)$. By Lemma 5.3.1 we have $e_1(r_1) >_2 e_4(r_0)$.

(II): By separation constraint we know that $s_2(r_l) >_2 e_2(r_1)$ and $e_5(r_0) >_2 s_2(r_r)$. By Lemma 5.3.1 we have $e_2(r_1) >_2 e_5(r_0)$. This is the same as for 2.II.

□

As a direct consequence of Theorem 5.3.5 we state the following corollary.

Corollary 5.3.5.1. *Let r_0, r_1, r_l, r_r be four regions, s.t. $\{r_0, r_1\} \in \mathcal{T}$, $(r_0, r_1) \in S_i$ and $r_l \in L^{min}, r_r \in R^{min}$. Then we can always draw a line parallel to z_j^o, s.t. $j \in \{0, 1, 2\} \backslash i$ between r_l and r_r.*

Existence of the Leader Line

We differentiate between two cases. In the first case, both of the intersections are empty. Then we can simply trace one of the areas upper/lower boundary and we have a leader line. Assume therefore that both S' and S'' are non-empty. Then in the other case, we assume we trace the lower boundary of S'. Tracing the upper boundary of S'' is equivalent up to symmetry.

If there exists a region r that intersects l at the vertical segment at the start of l, then $r \notin L^{min}$ since all regions in L^{min} are completely to the right of this segment. The vertical segment is only drawn if there exists a region $r_l \in L^{min}$, s.t., $s_1(r_0) >_1 l_4(r_l)$. This means that the starting point of l was excluded from A' and we need to move downwards first. If $(r, r_l) \in S_1 \cup S_2$, then r can not intersect l and by Lemma 5.3.5 $r \notin R^{min}$. This means the center of r must be placed either to the right of $s_0(r_1)$ or to the left of $s_0(r_0)$. If $(r_1, r) \in S_1$, $(r, r_1) \in S_2$ or $(r_0, r) \in S_2$ then r is separated from the segment by either r_0 or r_1. If $(r, r_0) \in S_1$, then r can actually intersect the vertical segment. But since we know through Lemma 5.3.2 that no region in L^{min} can be adjacent to r, we can simply trace the outline of r. It is of note that since r can only interfere with this vertical segment with the edge $e_1(r)$, the distance traveled is the same is if we would have traced the outline of A' or shorter. This special case is illustrated in Figure 5.12. The intersection for the last vertical segment can be argued similarly.

Assume r intersects l at a segment parallel to z_1^o. Then r can not be in L^{min} since l would have simply traced the outside of r. Some part of r would lie above the leader line, which makes a separating line between r and the hexagon responsible for the current bend parallel to z_1^o or z_2^o impossible and by Lemma 5.3.5.1 it can not be in R^{min}. If $(r_1, r) \in S_1$, $(r, r_1) \in S_2$ or $(r_0, r) \in S_2$ then r is separated from the segment by either r_0 or r_1. If $(r, r_0) \in S_1$, then l would have traced r as described above. The intersection with a segment parallel to z_2^o can be argued similarly.

Therefore we conclude that a leader line as constructed above always exists.

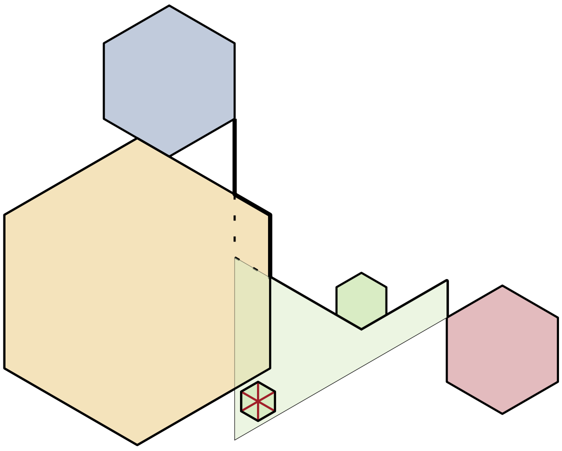


Figure 5.12: Illustration of a case, where the vertical segment is broken up. Since the placement of the green hexagon indicated with a red star is impossible, we never need to backtrack, and since the way, directly through the obstacle is as long as the way around, we stay at the same length.

Length of a Leader Line

In the case where both intersections are empty, we can draw a simple line between the two regions, which results in a lower bound of L.

$$L = \frac{2 \cdot k_0(r_0, r_1)}{\sqrt{3}}$$

If one or both intersections are non-empty, we might be forced to draw a vertical segment into the leader line, which increases the length.

The upper bound on the length of l is, as already argued above, the length of the boundary of A' or A''. However since we do not know how the separation constraints between the regions in L^{min}/R^{min} and r_0/r_1 are set up, we need to take the maximum of both values.

The distance $k_0(r_0, r_1)$ together with the two angles of 90° at r_0 and 30° defines a triangle that has a left edge of length $\frac{k_0(r_0,r_1)}{\sqrt{3}}$. When looking at the distance to the highest point of A'' a positive $\delta_0^o(r_0, r_1)$ adds to the total length of the distance, while a negative $\delta_0^o(r_0, r_1)$ subtracts from this length. And since the distances are measured from the center points, we need to subtract the weights of the hexagons. This leaves us with a distance

$$d^{high} = \max\left(\left|\frac{k_0(r_0, r_1)}{\sqrt{3}} + \delta_0^o(r_0, r_1)\right| - \frac{(w(r_0) + w(r_1))}{2}, 0\right)$$

When looking at the lowest point of A' we get a similar result, however we need to start out with the negative value of $k_0(r_0, r_1)/\sqrt{3}$ since the distance is traveled in negative z_0^o-direction and the additive effect of $\delta_0^o(r_0, r_1)$ is reversed.

$$d^{low} = \max\left(\left|-\frac{k_0(r_0, r_1)}{\sqrt{3}} + \delta_0^o(r_0, r_1)\right| - \frac{(w(r_0) + w(r_1))}{2}, 0\right)$$

With this we can finally give an upper bound B for the length of a leader line l.

$$B = max\left(d^{high}, d^{low}\right) + L$$

With this bound established, we state the following theorem.

Theorem 5.3.6. *Let r_0, r_1 be two regions, s.t. $\{r_0, r_1\} \in \mathcal{T}$ and let there be a hypothetical cartogram $\mathcal{A}$. The adjacency between two regions r_0, r_1 in a cartogram $\mathcal{B}$ can always be visualized with a leader line l of length $|l|$, s.t., $L \leq |l| \leq B$.*

CHAPTER 6

Implementation

This thesis we implemented the linear programs described in Section 4, including a graphical user interface which lets the user load an input map and associated values for the countries in different layers, choose the method which is used to create (multiple) stable cartograms, choose the settings of the chosen method, measure the generated maps with a set of metrics, visualize qualities of the maps and export the generated maps to a file.

6.1 Description

In this section we will describe the input formats, the extraction of the relevant details, preprocessing, the implementation details of the LP's, all implemented extensions, quality of life functionality and the interface of the applications.

6.1.1 Input Formats

The program reads the map as an *.ipe* file. The *.ipe* file format is part of "The Ipe extensible drawing editor". A definition of the format can be found in the Ipe manual [Che19]. The input file contains multiple polylines and text labels. Every border between countries or a country and a body of water is represented as a polyline. All participating countries of such a border are saved with the polyline. The boundary of every country c can be constructed by taking the union over all polylines which form borders of c. Every country is marked by a text label, which contains a three letter identifier and is placed inside the boundary of a country.

The data sets are read in semicolon separated *.csv* format. The file includes a header and every line specifies the name and country code of a region, as well as the value for this region in every cartogram. Every column therefore contains the sizes of all regions in a single cartogram, including a dummy column for the input layer. The values in this column are ignored. It is assumed that the data does not contain missing values.

6.1.2 Preprocessing

From the .ipe input file of the map, the implementation creates a *DCEL*. A DCEL is a "doubly connected edge list", a commonly used data structure created to handle queries and traversal on graphs. A detailed explanation of a DCEL can be found in any standard textbook; we refer to the lecture notes of Mount [Mou07]. In a DCEL every edge is represented by two darts, one of which is on either side of the edge and belongs to the corresponding face. By iterating over all darts of a face, we can therefore find all faces that share an edge with this face. From this we build an adjacency graph. At this stage we do not guarantee that the resulting adjacency graph is planar, as countries can consist of multiple disjoint polygons. This must be insured through the actual input map.

The center point of the bounding box of the polygon in the input map is used as the original position of a region. The distance between two regions is measured between these points. For a Demers cartogram we measure the x and y-distances between these points. The separation constraint for two regions r, r' is calculated as follows:

$$\text{If } |x(r) - x(r')| > |y(r) - y(r')| : \begin{cases} (r, r') \in S_x & \text{If } x(r) - x(r') < 0 \\ (r', r) \in S_x & \text{Othw.} \end{cases}$$

$$\text{If } |x(r) - x(r')| \leq |y(r) - y(r')| : \begin{cases} (r, r') \in S_y & \text{If } y(r) - y(r') < 0 \\ (r', r) \in S_y & \text{Othw.} \end{cases}$$

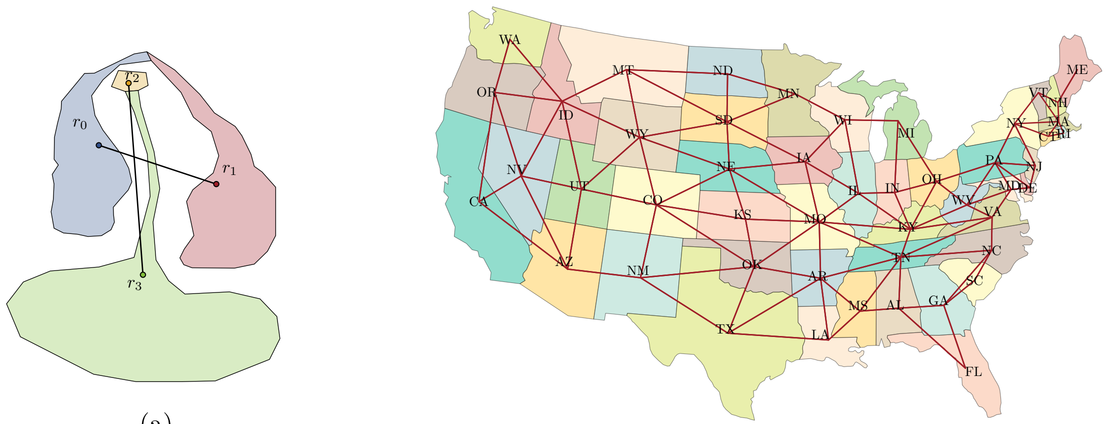
In an iso-hexagon cartogram we calculate the separation constraints similarly:

$$i' = \max_{|z_i(r) - z_i|} i \in \{0, 1, 2\}$$

$$\begin{cases} (r, r') \in S_{i'} & \text{If } z_{i'}(r) - z_{i'}(r') < 0 \\ (r', r) \in S_{i'} & \text{Othw.} \end{cases}$$

Note that this can lead to extreme combinations of separation constraints, which are not possible if the input is already a Demers cartogram or iso-hexagon cartogram. One example is Figure 6.1a, which would lead to a non-planar embedding of the adjacency graph. However many real life maps fulfill the criteria of an adjacency graph with a planar embedding. Figure 6.1b is an example of the United States with the corresponding planar embedding. This is also the map that was used for the evaluation of this system.

We also have to mention another caveat of this method. Since the primary separation constraint is enforced based on distance instead of the already existing separating line, a situation can arise in which two countries have a separating line in the input, however we enforce the orthogonal separating line in the output. This method is not optimal,



(a)

(b)

Figure 6.1: (a) An extreme configuration in which countries are interior disjoint, but the implied embedding of the adjacency graph connecting their centroids is not planar. (b) The map of the united states, which was used for the evaluation. Here the implied embedding is planar.

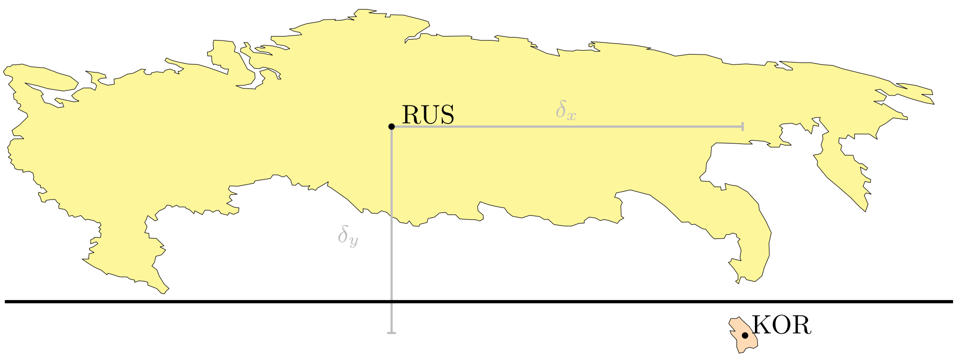


Figure 6.2: The y-distance between Russia and the Republic of Korea is smaller than their x-distance, however the input only contains a splitting line orthogonal to the y-axis. The LP enforces the existence of a splitting line orthogonal to the x-axis, in the strong setting additionally a splitting line orthogonal to the y-axis between the regions.

as it does not always match up with the perceived relative placement of countries. An example of this situation is Figure 6.2, in which Russia and the Republic of Korea have a separating line orthogonal to the y-dimension in the input and our method enforces a separating line orthogonal to the x-dimension, despite not being present in the input.

By definition two regions have only a single separation constraint. If $(r, r') \in S_x$ then we know that we enforce a separating line between r and r', orthogonal to the x-axis. The implementation of the strong setting of the Demers cartograms uses again the bounding boxes of the input polygons and checks if there exists a separating line between the two regions orthogonal to the y-axis. If this is the case, then we will also enforce the existence of a second separating line in the output cartograms, by simply adding the separation constraint for both dimensions.

The data is read in and saved for every region. We then normalize the data sets. Depending on the semantic relation between the data sets we go about the normalization in a different fashion. If the data sets are not related, e.g., if we want to visualize three cartograms of the GDP per capita, the forest area and the average age of the prime minister, we will encounter vastly different numbers. If we would take these numbers as the area of the regions in the cartograms directly, we would end up with either a gigantic, or a tiny cartogram. In order to handle this, we scale the data to a sensible minimal and maximal size, s.t., the relative size differences in area value are kept intact. If we however use time/series data sets, e.g., data sets which depict the same kind of data at different points in time, we do not want all cartograms to be scaled to the maximum and minimum value individually, because a trend of growing (or shrinking) maximal (or minimal) values would not be visible in this case. We rather want to normalize these data sets as a whole, in order to ensure comparability between regions of different cartograms. If we encounter a data set with related and unrelated sets of columns, we can normalize each set as one entity. This can be controlled with a command line argument as described in Section 6.1.4.

As we interpret the data value as the desired area of the polygon which represents each country, and the actual saved weight of a region is interpreted by the program as the edge length of the polygon, we make a distinction between a Demers cartogram and a iso-hexagon cartogram. Let $w_n(r)$ be the (normalized) data value of region r in the data set and let $w_{dem}(r)/w_{hex}$ be the weights of the region which are saved inside the program.

$$w_{dem} = \sqrt{w_n}$$

$$w_{hex} = \sqrt{\frac{2 \cdot w_n}{3\sqrt{3}}}$$

6.1.3 LP Implementation

The main part of this implementation is the implementation of the LP's. All constraints were implemented as they were defined in Chapter 4 using Java 8 and the linear program solver IBM ILOG CPLEX Optimization Studio. In the following we will refer to these constraints as follows:

- The constraint ensuring disjointness between polygons, by enforcing a minimal distance between them is called the *first constraint* or *disjointness constraint*.
- The constraint that adds variables that enable the LP to minimize the distance between adjacent regions will be called the *second constraint* or *adjacency constraint*.

- The constraint that enables the LP to determine a unique optimal solution in the case of adjacent regions by minimizing the discrepancy of the slope in the input and output will be called the *third constraint* or *nuance constraint*.
- The constraint that adds all variables to the LP, which are necessary to minimize the distance between the original and the actual placement of a region will be called *fourth constraint* or *origin constraint*.
- The constraint that adds the possibility of minimizing the distance between the placements of regions in different cartograms will be called the *fifth constraint* or *stability constraint*.

Every constraint is implemented in a way that it can be turned on or off, i.e., all variables and (in)equalities needed for this constraint are either added to the LP or not. The objective function minimizes only variables for which the constraints are enabled. The information about enabled constraints is maintained as a boolean array `constraint_controls`. If `constraint_controls[i]` is set to true, then the i-th constraint is set to true. The stability controls, i.e., between which cartograms we minimize the fifth constraint, are again implemented as described in Section 4.1.5. In the complete and successive stability model, one LP for all cartograms is created, solved and then used to extract the positioning of the regions in every cartogram. In the iterative stability model, we create an LP for a single cartogram, solve it, extract the positioning in this first cartogram and then use this placement as a constant value while instantiating the LP for the next cartogram.

6.1.4 Interface

In this section we describe the interface of the implemented application.

Command Line Arguments

The behavior of the application can be controlled by the means of command line arguments. Below we present a list of possible arguments, their usage and their behavior

- (-out "PATH/") sets the output path for the produced cartograms and quality measures. The results are exported as an .ipe file. The quality measures are exported to a .json file.
- (-ipe "PATH/map.ipe") sets the input .ipe file path.
- (-wgh "PATH/weights.csv") sets the data .csv file path.
- (-min "o/d/c") controls the minimization goal. The three presets differentiate between minimizing the distance of a region to its original placement, minimizing the distance between regions that should be adjacent and the number of lost

adjacencies, by setting the first four constraint controls accordingly. The fifth control, concerning the stability model, is set by a different flag.

- "o" sets the constraint controls to `{true, false, false, true, ______}`
- "d" sets the constraint controls to `{true, true, false, true, ______}`

- (-opt "co/su/it") sets the stability model of the implementation to either the complete (co), the successive (su) or the iterative model (it).
- (-nostab) disables the stability constraint altogether by setting the constraint controls to `{______ ______ ______ ______ false}`. If this flag is not set, the constraint controls are `{______ ______ ______ ______ false}` by default.
- (-hex) switches the application from creating Demers cartograms to creating iso-hexagon cartograms.
- (-a) enables the automatic mode which runs loads the map and data, creates the LP model, solves it and writes the results to the specified output path

6.2 Experiments

In this section we describe the experimental setup and the results of the evaluation.

6.2.1 Setup

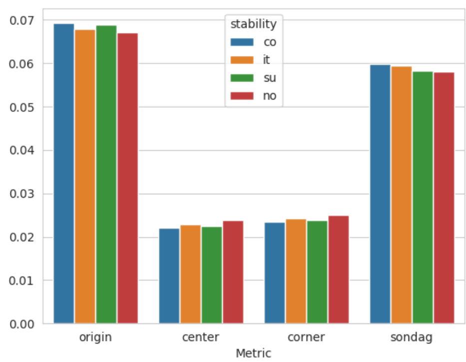
The experiments were run on a computing cluster with 16 nodes, each with 160GB RAM and two 10-core Intel Xeon E5-2640 v4, 2.40GHz (i.e., 20 cores per node). The system runs a Linux system with a kernel version 4.15.0. Our implementation uses IBM ILOG CPLEX 12.8 to solve (integer) linear programs, each running on a single thread. Every experiment had an allocated available memory space of 8GB. The produced logfiles (in .json format) were analyzed, evaluated and visualized in Python 3.6.8, using the python libraries pandas (0.24.2), numpy (1.16.4), json (2.0.9), matplotlib (3.1.1) and seaborn (0.9.0).

All experiments were performed using the a modified map of the United States. This map contains all US states excluding Hawaii and Alaska, and the following adaptions were performed. Michigan, which consists of two big separate landmasses, one on each side of Lake Michigan was connected up to consist of only a single polygon. All en- and exclaves were removed as well as all islands. The territory of the reservations is counted as part of the containing state. The area of Washington DC is counted as part of Maryland. The map is displayed in Figure 6.1b.

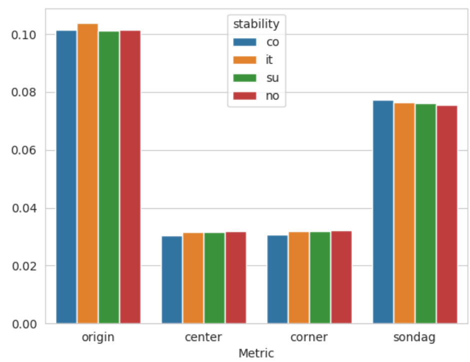
For the experiments, we use four time series, of different (but overlapping) temporal ranges, all of which are listed in Table 6.1. The Drug Poisoning Mortality data set is from the Center for Disease Control and Prevention [Cen19]; the General Election Turnout data set is from the United States Elections Project [McD19]; the GDP data set is from

Map	Time series	Years
US	Drug Poisoning Mortality	2007–2016
US	GDP	2007–2016
US	General Election Turnout	1998–2016 (even years)
US	Population	2011–2020

Table 6.1: Time-series data sets used in our experiments.



(a) Demers



(b) Iso-Hexagon

Figure 6.3: Stability and quality measures averages over all created Demers cartograms (a) and iso-hexagon cartograms (b) categorized by stability model.

the US Bureau of Economic Analysis [U.S19]; and the US Population data set is from the US Census Bureau [U.S18]. From all data sets, we produced a fifth data set, which contains one entry from each of the other data sets all in the same year. This data set can therefore not be classified as time series as the data comes from the same year, but has not necessarily a semantic connection.

All data were normalized to range with a specified minimum and maximum size for the squares/hexagons. The sizes are dependent on the diagonal Δ of the input bounding box, i.e., the size of the smallest axis-aligned rectangle that contains all bounding boxes of the input polygons. The maximum size of a rectangle is 30Δ while the minimum size is set to $\frac{\Delta}{20}$.

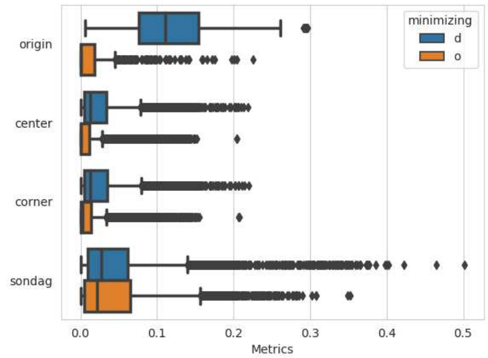
6.2.2 Results

The first distinction we want to look at is the impact of the chosen stability model on the evaluation results of the produced cartograms. For this we measured the average center and corner travel distance, number of lost adjacencies, origin displacement and stability score after Sondag et al. over all cartograms produced with the same stability model.

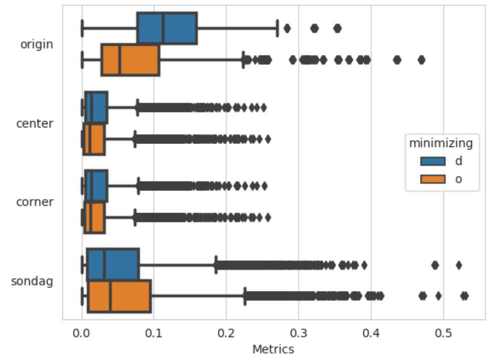
Table 6.2 shows the number of lost adjacencies, while Figure 6.3 shows the four mea-

Stability Model	Demers cartograms	iso-hexagon cartograms
complete	62.680	56.880
iterative	61.900	56.580
successive	62.795	56.335
no model	61.095	56.000

Table 6.2: Average number of lost adjacencies over all created Demers cartograms and iso-hexagon cartograms categorized by stability model.



(a) Demers



(b) Iso-Hexagon

Figure 6.4: Box plot of stability and quality measures averages over all created Demers cartograms (a) and iso-hexagon cartograms (b) categorized by minimization model.

sures. It can clearly be seen that the differences between the four different possible stability models (complete, successive, iterative, no stability constraint) is negligible. The conclusion which we draw from this is that the restriction imposed by the other enabled constraints are already resulting in considerably stable cartogram layouts. We will restrict our analysis on the basis of this observation to the case in which we do not enable any stability constraint.

The next comparison is between the different methods of optimizing a cartogram. We run our experiments with two different settings, i.e., minimizing the distance between the original location and the placement in a given cartogram and minimizing the distance between regions which ought to be adjacent. We call these two methods the *origin model* (o) and *distance model* (d). Again we accumulated all runs of the same model and plotted their averages as well as box plots for the four taken measures, displayed in Figure 6.4. Runtimes and lost adjacencies are reported in Table 6.3.

We can see some differences between the two models, however before reporting on these differences we want to point out that the origin model of minimization is dependent on the scaling of the data. Extremely small maximal values would result in low (possibly no) overlap. This enables every region to be placed at its initial position without being

Type (minimization model)	# lost adjacency	Solve time of LP (in seconds)
Demers cartograms (d)	47.00	3.066
Demers cartograms (o)	76.55	2.501
iso-hexagon cartograms (d)	56.50	7.133
iso-hexagon cartograms (o)	56.25	2.566

Table 6.3: Average number of lost adjacencies over all created Demers cartograms and iso-hexagon cartograms categorized by stability model.

moved. Extremely large scaling on the other hand would result in necessarily big values of displacement from the origin simply to fulfill disjointness. Note that the method of minimizing distances between supposedly adjacent distances is not similarly dependent on the extent of the scaling.

With this in mind we turn to the discrepancies between the Demers cartogram and iso-hexagon cartogram experiments. While we can see a lower value in origin displacement for the origin model in both cases, which is expected as this is the primary minimization goal of the model, the difference observed in the Demers case is significantly bigger than in the iso-hexagon case. At the same time we can see no difference in the number of lost adjacencies in the iso-hexagon cartograms, while the Demers cartograms show a sizable gap of almost 30. From this we conclude that the square shape of the regions in combination with the grid like topography of the US states enabled more states to be placed initially at their origin without causing overlap and therefore without reason to move and to create contact with adjacent regions. This is an example of the scaling dependency of the origin model.

The measures taken are a try to capture the quality of a cartogram with a metric. It is however irreplaceable to compare the created maps themselves visually, since a measurement can only go so far in trying to capture the resemblance of a map to its original topology. We will therefore present on the next pages some of the created maps. We present Demers cartograms and iso-hexagon cartograms for the first and third data set (see Table 6.2). All cartograms were created using no stability constraint and either the origin or the distance model.

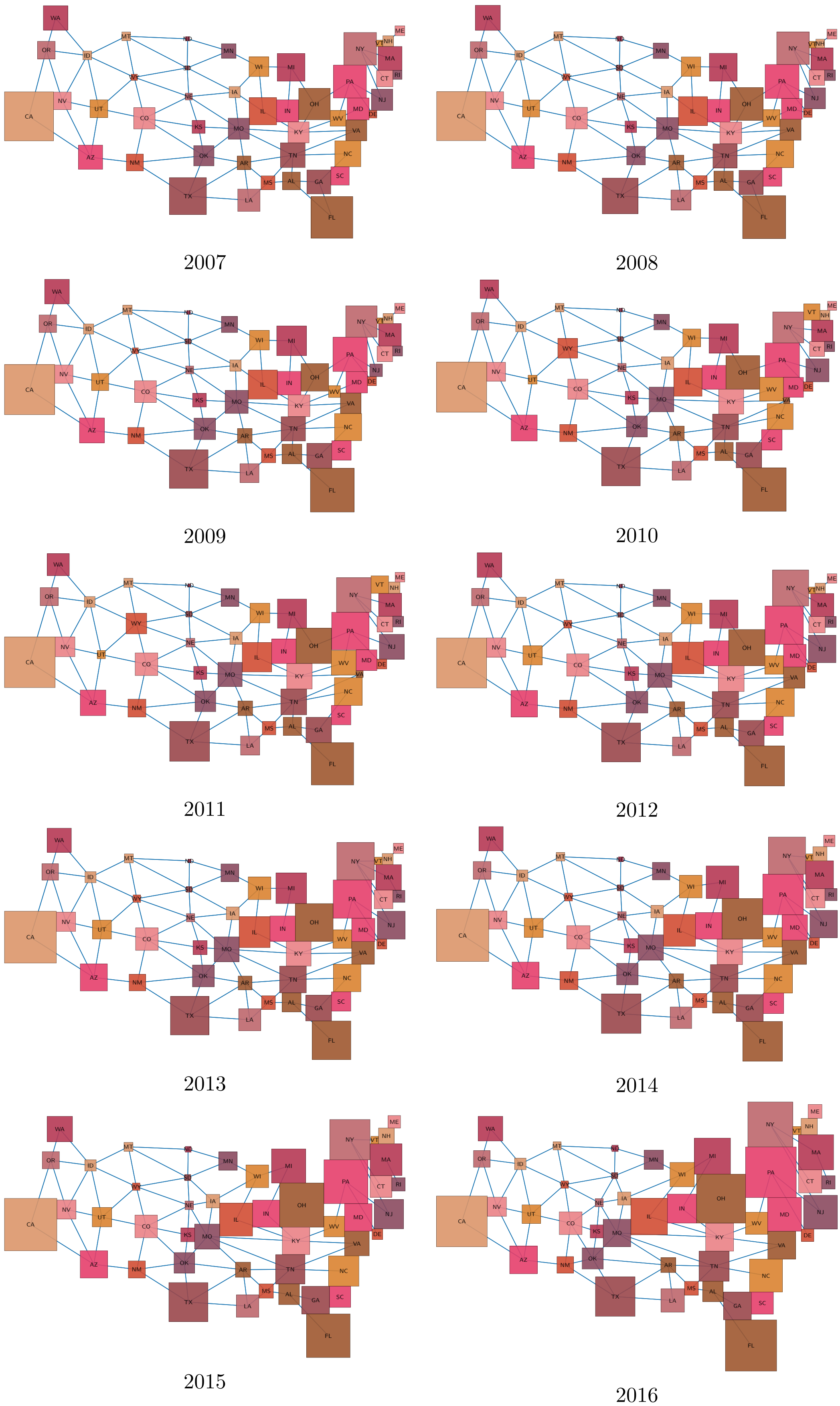


Figure 6.5: Demers cartograms of the drug mortality rate in the US by state. Created using the origin model, without stability constraint.

Figure 6.6: Demers cartograms of the drug mortality rate in the US by state. Created using the distance model, without stability constraint.

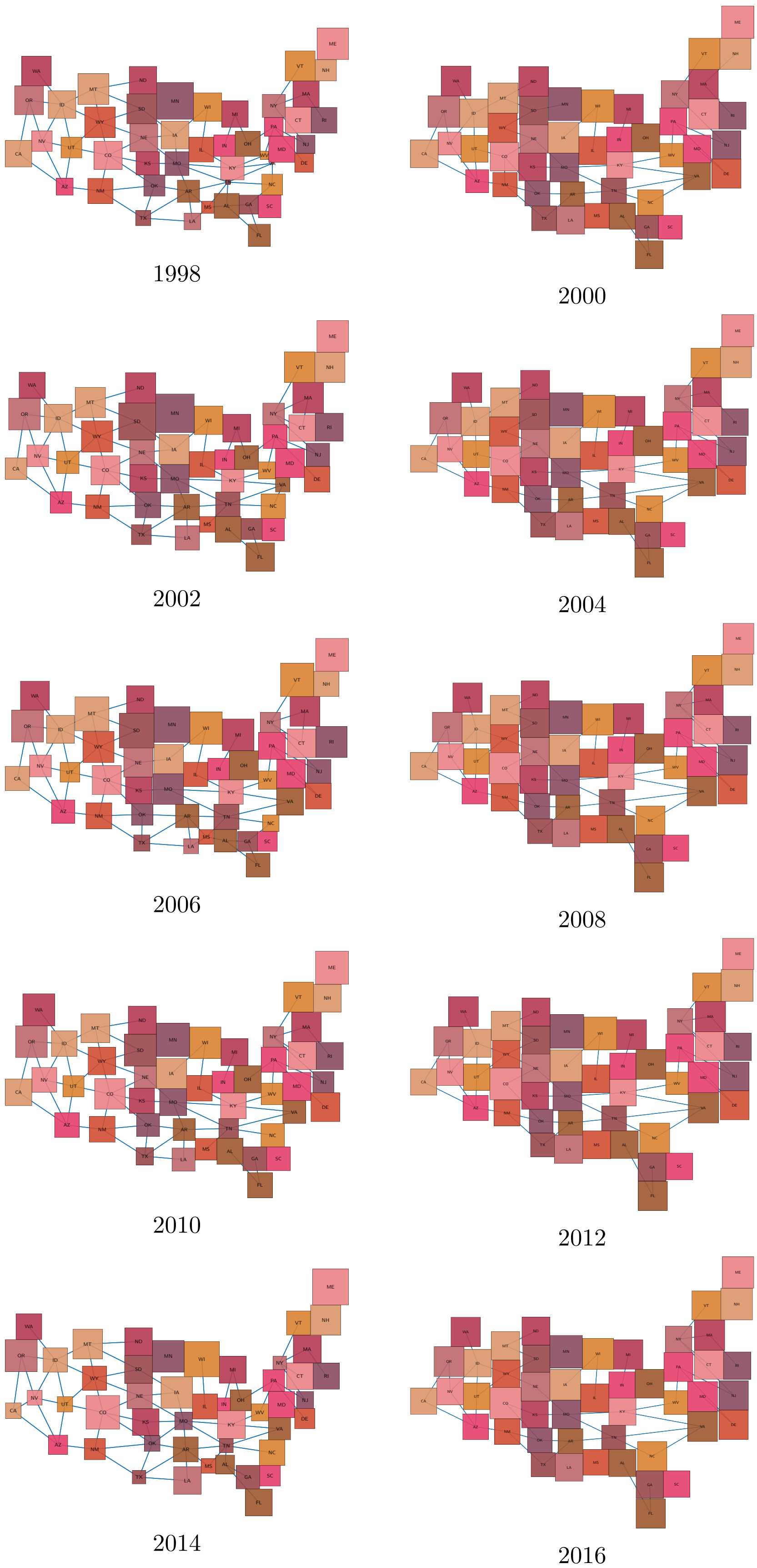


Figure 6.7: Demers cartograms of the general election turnout in the US by state. Created using the origin model, without stability constraint.

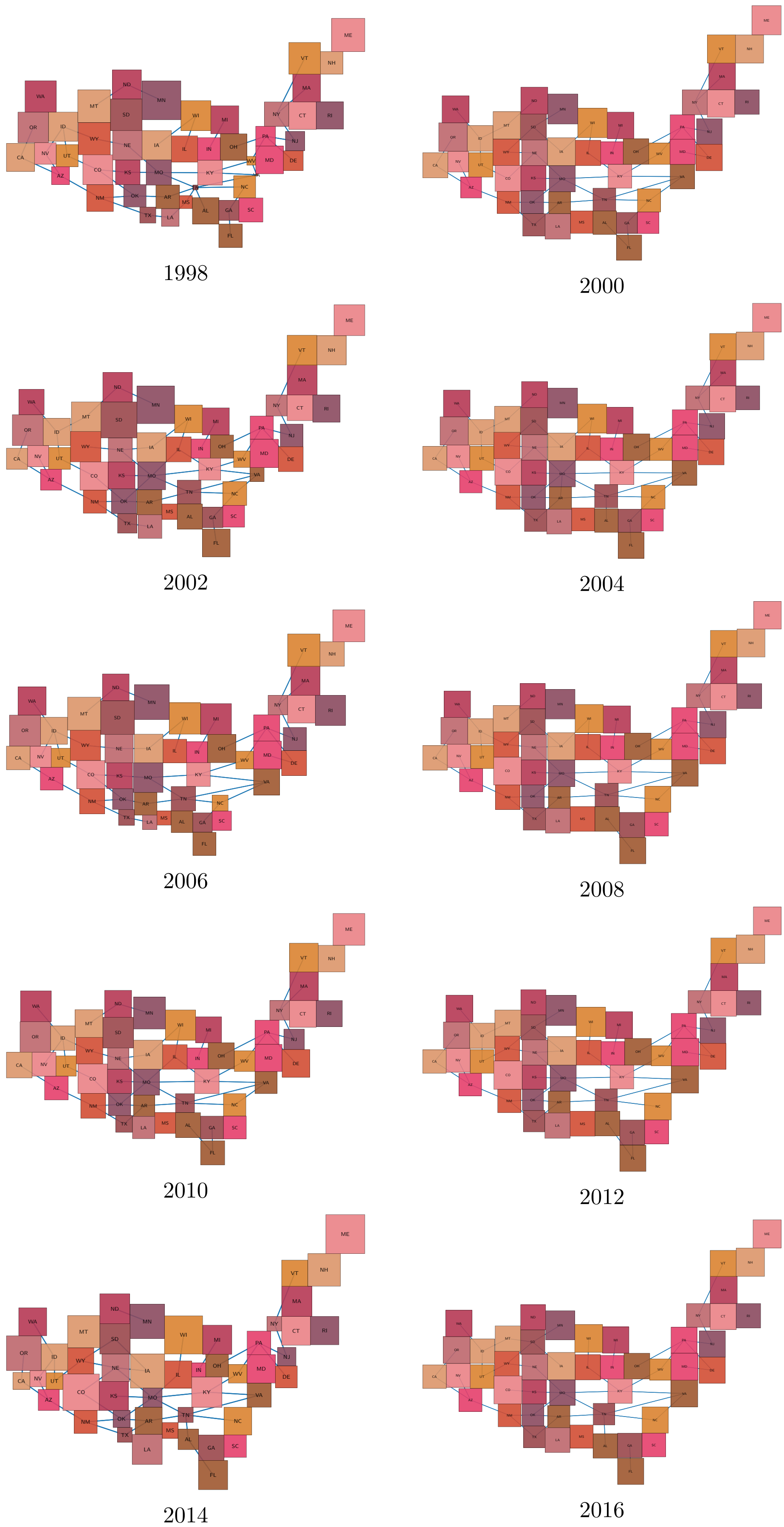


Figure 6.8: Demers cartograms of the general election turnout in the US by state. Created using the distance model, without stability constraint.

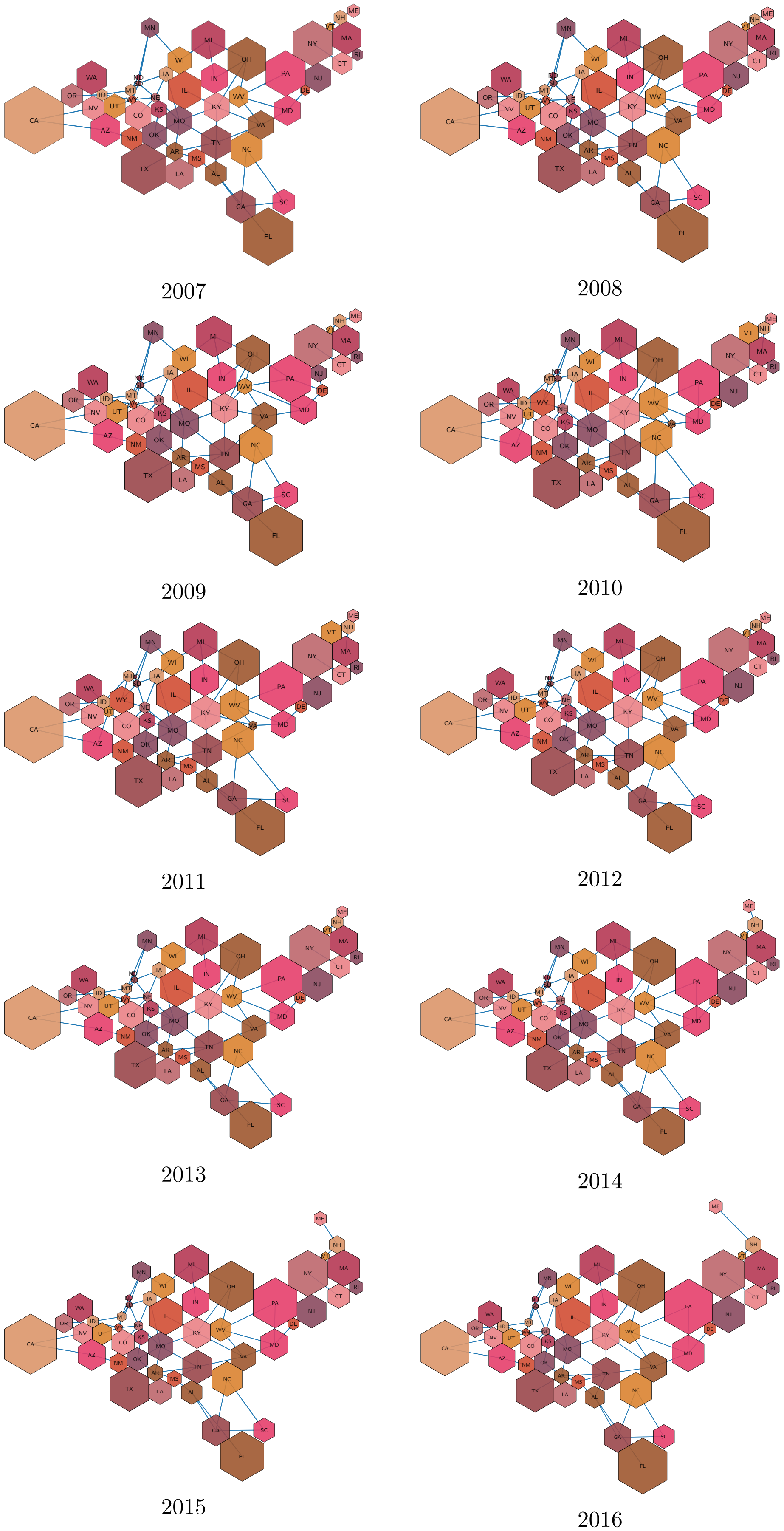


Figure 6.9: Iso-hexagon cartograms of the drug mortality rate in the US by state. Created using the origin model, without stability constraint.

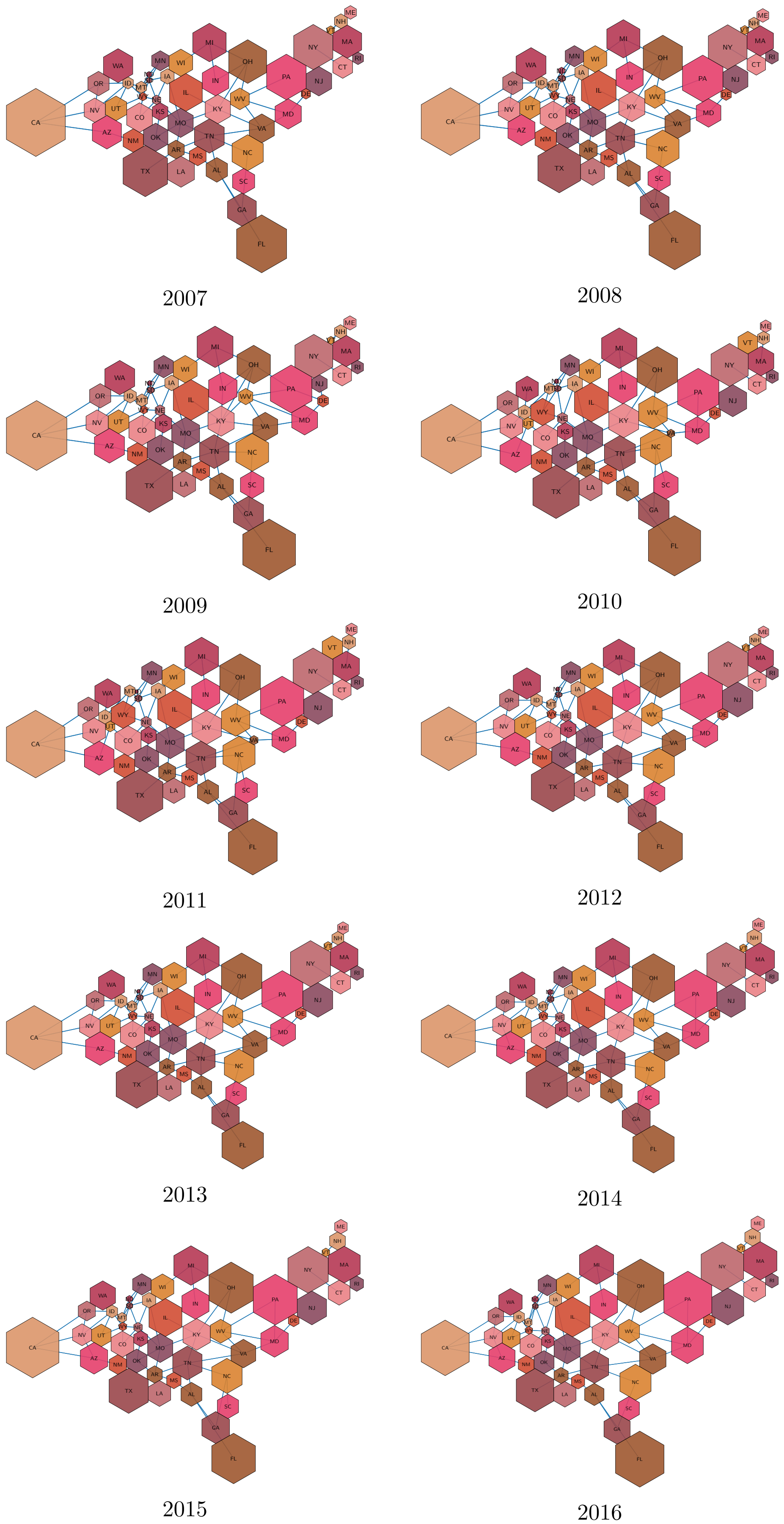


Figure 6.10: Iso-hexagon cartograms of the drug mortality rate in the US by state. Created using the distance model, without stability constraint.

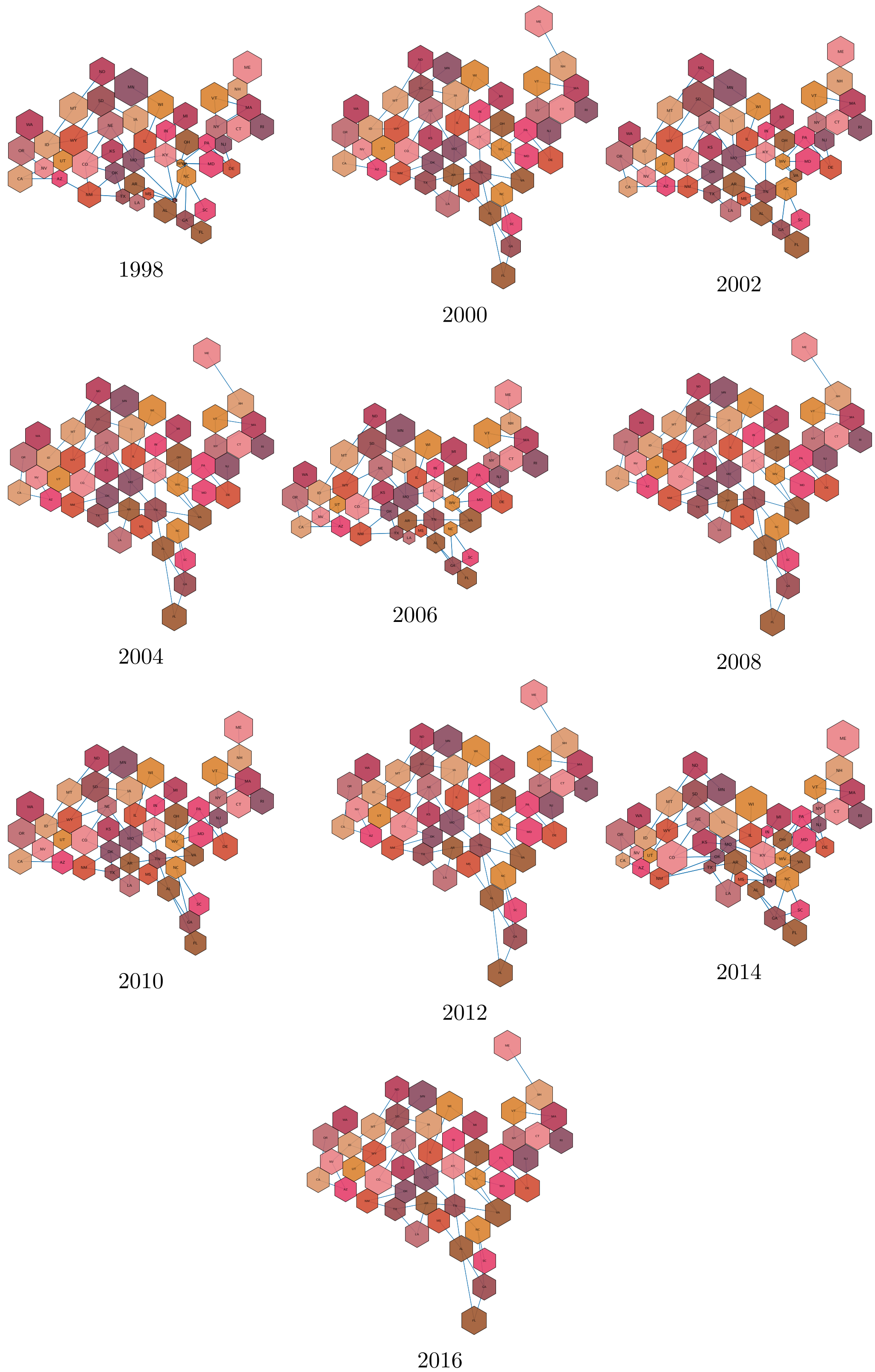


Figure 6.11: Iso-hexagon cartograms of the general election turnout in the US by state. Created using the origin model, without stability constraint.

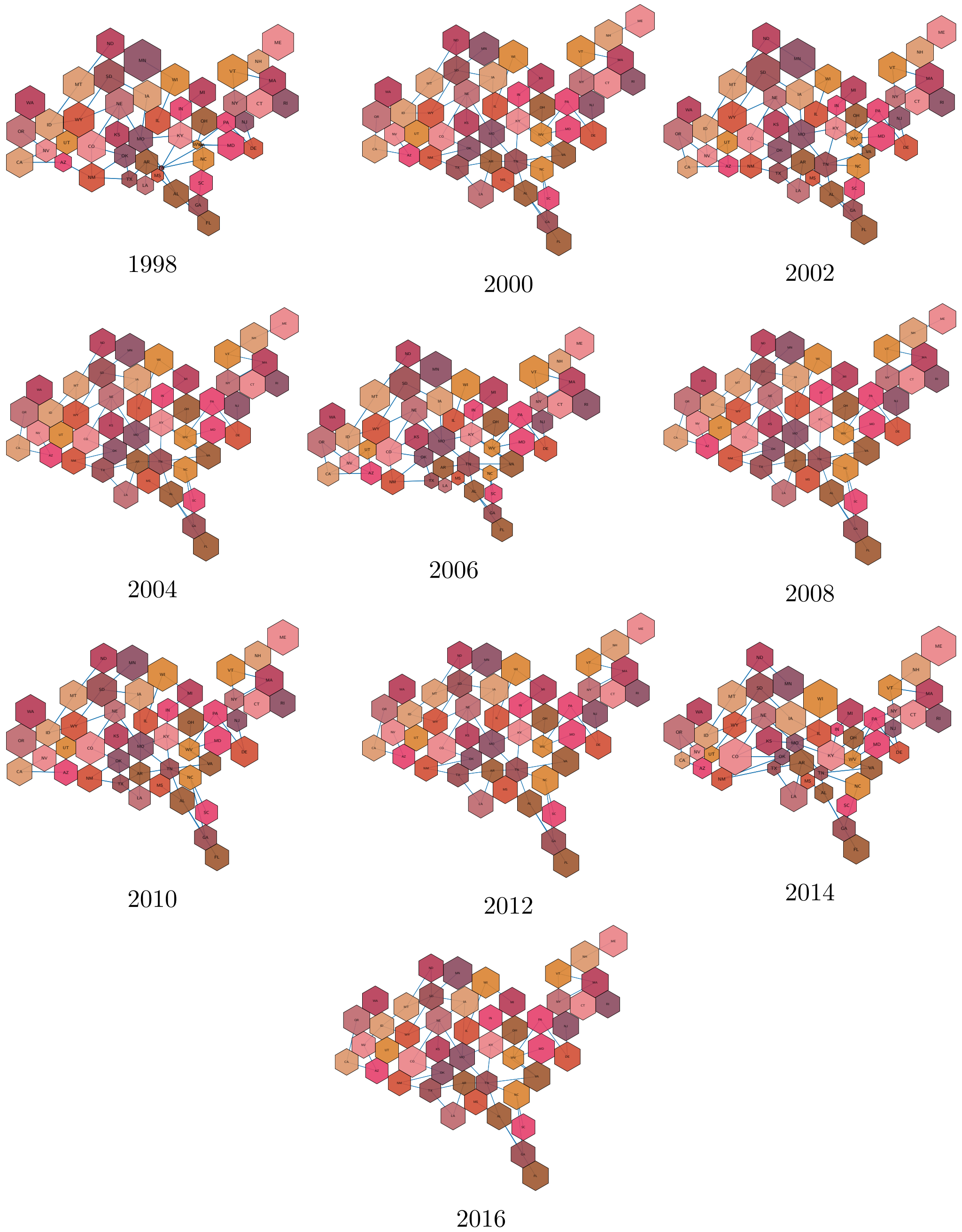


Figure 6.12: Iso-hexagon cartograms of the general election turnout in the US by state. Created using the distance model, without stability constraint.

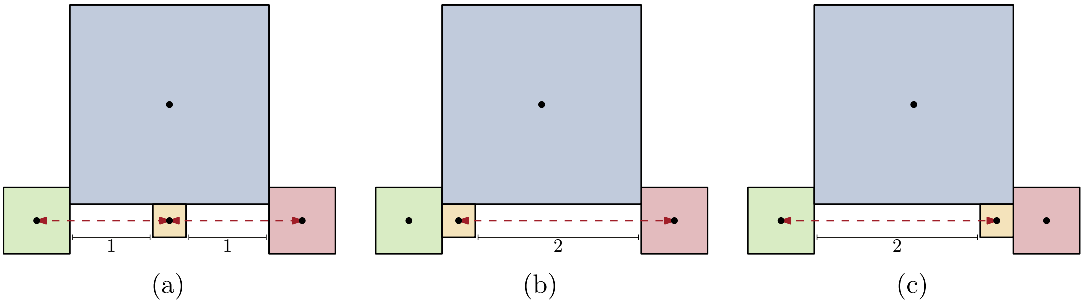


Figure 6.13: Configuration of squares which are indistinguishable in quality by simply minimizing the distance between adjacent regions. A red arrow between squares indicates a lost adjacencies. In all three cases the sum of the distance between adjacent regions is 2, however in cases (b) and (c) only one adjacency is lost.

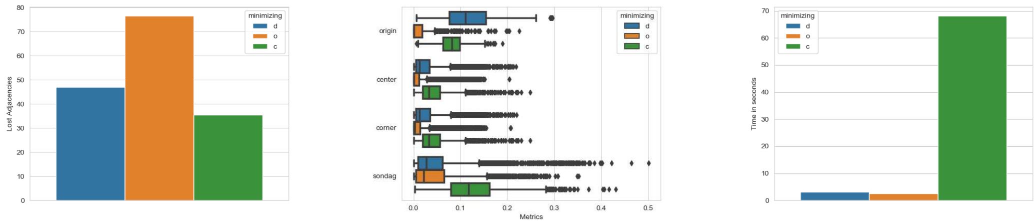
6.3 Proposed Extension

A already mentioned, minimization of the distance between adjacent regions is not sufficient to guarantee a minimum of lost adjacencies. Assume the existence of four regions as depicted in Figure 6.13. The sum of distances between adjacent regions is 2 in all three cases, even though in the first case, we lose one adjacency more than in the other two.

An approach to handle this problem can be to introduce a boolean variable d_c into the linear program, i.e., a variable with value either 0 or 1. We will call this approach the *count model*. This changes the linear program into a mixed integer program, as it contains real and integer number variables. We will sketch the theoretical idea behind this approach for the Demers cartograms, but it can easily be adapted to the iso-hexagon cartograms. Let δ be the minimal distance that two regions r_1, r_2 must have due to their size and the disjointness constraint. Let $(r_1, r_2) \in S_x$. Then we want the differences $|x(r_1) - x(r_2)|$ and $|y(r_1) - y(r_2)|$ to be smaller or equal to δ. In fact $x(r_1) - x(r_2)$ needs to be exactly δ as that is its minimal value. If this is not the case, we will subtract sufficiently big constant C_c, which is chosen such that the displacement can not possibly be bigger than this constant. This can for instance be the sum of all widths over all regions plus the minimal distance between non adjacent regions multiplied by the number of regions. Whether C_c is subtracted or not is controlled by setting the value of d_c. The constraint is as follows

$$|x(r_1) - x(r_2)| - \frac{w(r_1) + w(r_2)}{2} - (d_c^{r_1,r_2} \cdot C_c) \leq 0$$
$$|y(r_1) - y(r_2)| - \frac{w(r_1) + w(r_2)}{2} - (d_c^{r_1,r_2} \cdot C_c) \leq 0$$

The number of times, we need to set the variable d_c to 1 corresponds to the number of lost adjacencies. We can therefore minimize the sum over all d_c variables.



(a) (b) (c)

Figure 6.14: Measured average lost adjacencies (a), stability (b) and runtimes of the MIP in comparison to the distance and origin model.

$$\min \sum_{\{r_1,r_2\}\in\mathcal{T}} d_c^{r_1,r_2}$$

We performed exploratory experiments on two data sets—the data set containing information about the drug poisoning mortality in the years 2007 to 2016 and the mixed data set—and results show that minimizing the number of lost adjacencies leads to cartograms with a lower number of lost adjacencies (see Figure 6.14a). However the stability measures show mostly worse results than the other methods of optimization. The stability score of Sondag et al. shows significantly higher instability, while corner and center movement are slightly worse. The origin displacement is slightly better than compared to the distance model. Stability measures are displayed in Figure 6.14b. As already mentioned in Chapter 2, introducing boolean variables into the program can have an immense effect on the runtime. This also occurs in our experiments. As can be seen in Figure 6.14c, the runtimes increased by one order of magnitude. It is to be expected that if this method is applied to a bigger map, e.g., that runtimes could easily exceed multiple hours.

Finally we present on the next pages the cartograms that show the drug mortality rate in the US between 2007 and 2016 and were created using no stability constraints and the count model.

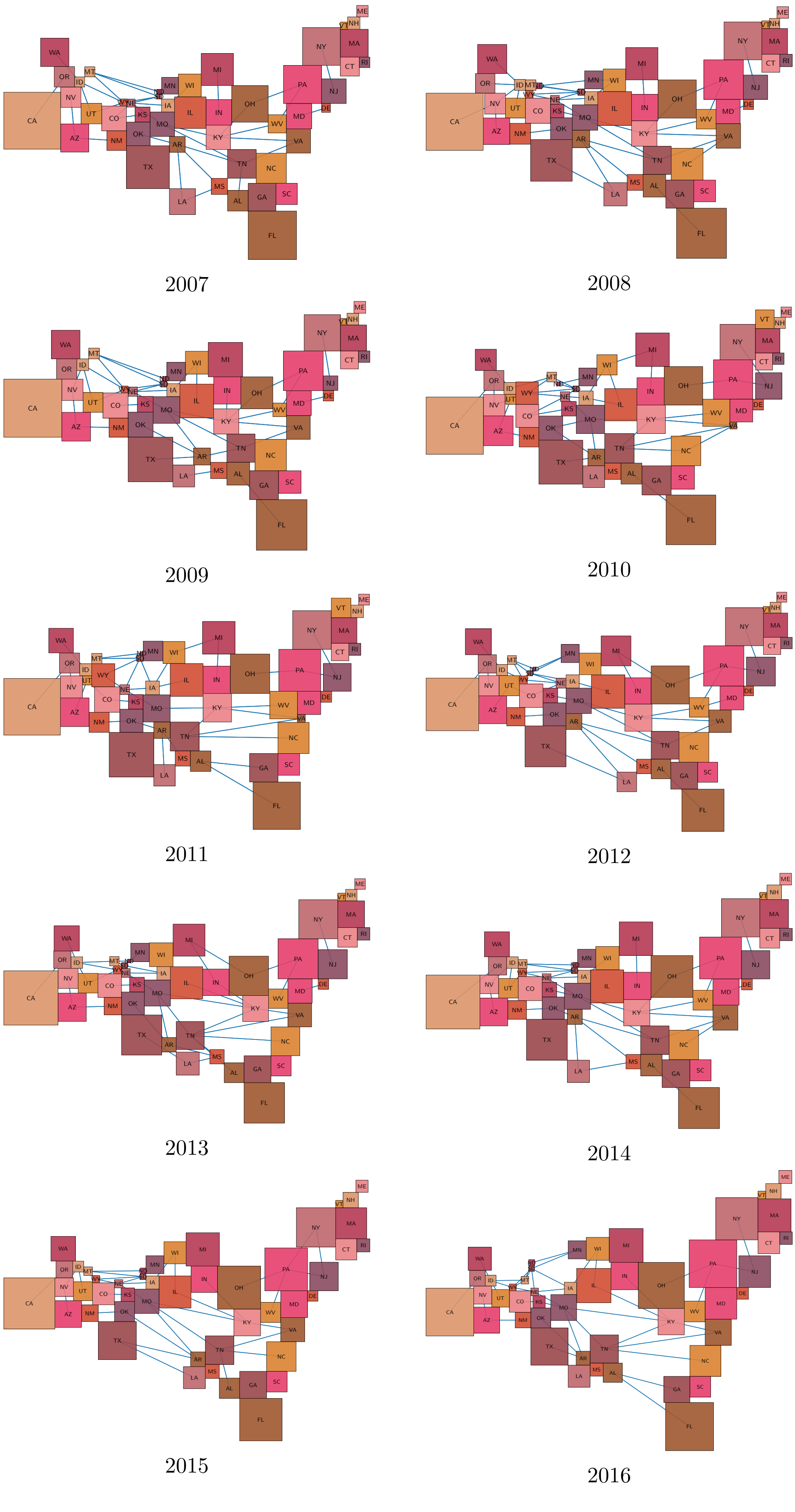


Figure 6.15: Demers cartograms of the drug mortality rate in the US by state. Created using the count model, without stability constraint.

CHAPTER 7

Conclusion

As we have shown in this thesis, there are multiple ways to approach the creation of a cartogram, even when using the same method of creation. We have presented a method of creating stable cartograms by enforcing separational constraints, which keep relative directions between regions. We have further proven the NP-hardness of generalized problems which do not enforce such separational constraints. Our experimental results suggest that enforcing the separational constraints results in reasonably stable cartograms.

There are several open questions left to answer. On the theoretic side, we built on already existing knowledge of NP-hardness and completeness for certain problems. We have seen that the deciding whether a graph has a proper square contact representation is NP-complete, as is deciding whether a graph has a hexagon contact representation. These were simply generalized cases of the actual underlying problem of deciding whether a given graph has a proper square contact or a hexagon contact representation that satisfies a set of separation constraints. The hardness of these questions remains open. Moreover the question of the existence of contact representations of a given graph with any given set of (possibly different) regular polygons is also unanswered.

Obviously the extension from Demers cartograms to iso-hexagon cartograms can be followed to its logical generalization of iso-k-gon cartograms, for which we can ask all the same questions. Excluding this generalization, there are common features of maps, which are notably absent in a visualization with a Demers cartogram or a iso-hexagon cartogram. A number of countries, consist of a collection of larger islands, a topographical feature, which is not at all represented in our version of these cartograms. The same is true for en- and exclaves. Relationships between different continents are also of interest as the inherent distance of countries separated by an ocean is an important geographical feature, which might be lost in our cartograms. There are multiple ways of handling this particular feature, e.g., by trying to minimize diversion from a desired distance for non-adjacent regions instead of simply minimizing distance between adjacent ones.

Alternatively, big bodies of water could be represented by dummy regions of desired size, which are then omitted in the final cartogram.

We have seen that certain scenarios force adjacencies to be lost. We presented visual aid to identify these lost adjacencies in the form of the leader lines and gave an upper bound on their length. It might be of interest to identify a similar bound for the complexity of such a leader line, if possible. Further under the assumption of additional restrictions, e.g., bounding the maximal number of neighbors for a given region, different options of visualizing the adjacency might be attractive.

In our evaluation section we have mentioned that origin model has the inherent flaw of being dependent on the scaling of the regions, even though it seems to be advantageous, when judges by its stability. In a visual comparison, we claimed that the overall shape of the underlying map seems to be better replicated by the distance model. It would be desirable to identify an additional measure which captures this replication of the underlying topology.

The model presented in this thesis tried to better stability with an additional constraint which turned out to underperform. We concluded that the additional stability gained from this constraint is negligible in comparison to the already existing stability of cartograms which fulfill the separation constraints. There might however be different approaches to guarantee stability altogether. Also there are several assumptions made in this thesis which can be approach a different way. To name just one, the decision made about the separation constraint could be improved with one of the mentioned methods of determining the relative placement of countries. The experimental result that the enforcement of the stability constraint has no significant impact on the quality of the generated maps must also be considered with the fact in mind that the used data sets were time-series data sets and the changes from one layout to the next in these data sets are not as radical as if we would compare unrelated data sets. Moreover the underlying map of the United States is expected to lend itself nicely to Demers cartograms and iso-hexagon cartograms due to its grid-like structure. Further analysis and evaluation with different maps and data sets is necessary.

We hope to see other results in this area and aim to extend our results in further work.

List of Figures

List of Tables

Bibliography

[BD11] Dimitris Ballas and Danny Dorling. Human scaled visualisations and society. *Handbook of GIS & society research*, pages 177–201, 2011.

[BDC02] Ian Bortins, Steve Demers, and Keith Clarke. Cartogram types. "`http://www.ncgia.ucsb.edu/projects/Cartogram_Central/types.html`", 2002. Accessed on 01.07.2019.

[BEF+12] David Bremner, William Evans, Fabrizio Frati, Laurie Heyer, Stephen G Kobourov, William J Lenhart, Giuseppe Liotta, David Rappaport, and Sue H Whitesides. On representing graphs by touching cuboids. In *International Symposium on Graph Drawing*, pages 187–198. Springer, 2012.

[BKS+11] Kevin Buchin, Vincent Kusters, Bettina Speckmann, Frank Staals, and Bogdan Vasilescu. A splitting line model for directional relations. In *19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 142–151. ACM, 2011.

[BT97] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[Cen19] Centers for Disease Control and Prevention. Drug poisoning mortality by state. `https://catalog.data.gov/dataset/drug-poisoning-mortality-by-state-united-states`, Mar 2019.

[Che19] Otfried Cheong. The ipe manual. `http://ipe.otfried.org/manual/onepage.html#sec:copyright`, 2019. Accessed on 02.07.2019.

[CKM+18] Markus Chimani, Stephen Kobourov, Wouter Meulemans, Martin Nöllenburg, and Jaakko Peltonen. Reimagining the mental map and drawing stability. In *NII Shonan Meeting Report*, 2018.

[Coo71] Stephen A Cook. The complexity of theorem-proving procedures. In *Third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[CR10a] Vijay Chandru and M Rammohan Rao. *Linear programming*, chapter 31, pages 31/1–31/37. 2010.

[CR10b] Vijay Chandru and M Rammohan Rao. *Linear programming*, chapter 32, pages 32/1–32/45. 2010.

[Dan51] George B Dantzig. Maximization of a linear function of variables subject to linear inequalities. *Activity analysis of production and allocation*, 13:339–347, 1951.

[DCN85] James A Dougenik, Nicholas R Chrisman, and Duane R Niemeyer. An algorithm to construct continuous area cartograms. *The Professional Geographer*, 37(1):75–81, 1985.

[DGH+11] Christian Duncan, Emden Gansner, Yifan Hu, Michael Kaufmann, and Stephen Kobourov. Optimal polygonal representation of planar graphs. *Algorithmica*, 63(3):672–691, May 2011.

[Dor96] Daniel Dorling. Area cartograms: Their use and creation, vol. 59 of concepts and techniques in modern geography. *University of East Anglia: Environmental Publications*, 2, 1996.

[Dor12] Danny Dorling. How to visualise social structure. "https://www.theguardian.com/news/datablog/2012/jul/24/danny-dorling-visualise-social-structure", 2012. Accessed on 06.07.2019.

[EW97] Herbert Edelsbrunner and Roman Waupotitsch. A combinatorial approach to cartograms. *Computational Geometry*, 7(5-6):343–360, 1997.

[FHQ08] Hannah Fairfield, Elaine He, and Kevin Quealy. What your global neighbors are buying. "https://archive.nytimes.com/www.nytimes.com/interactive/2008/09/04/business/20080907-metrics-graphic.html", 2008. Accessed on 06.07.2019.

[Fri74] Herman R Friis. Statistical cartography in the united states prior to 1870 and the role of joseph cg kennedy and the us census office. *The American Cartographer*, 1(2):131–157, 1974.

[Fun37] Howard Gray Funkhouser. Historical development of the graphical representation of statistical data. *Osiris*, 3:269–404, 1937.

[Giv16] Chris Givens. Demers cartogram. "https://bl.ocks.org/cmgiven/9d6bc46cf586738458c13dd2b5dadd84", 2016. Accessed on 16.07.2019.

[GN04] Michael T Gastner and Mark EJ Newman. Diffusion-based method for producing density-equalizing maps. *Proceedings of the National Academy of Sciences*, 101(20):7499–7504, 2004.

[HK98] Donald H House and Christopher J Kocmoud. Continuous cartogram construction. In *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pages 197–204. IEEE, 1998.

[HK01] Petr Hliněnỳ and Jan Kratochvıl. Representing graphs by disks and balls (a survey of recognition-complexity results). *Discrete Mathematics*, 229(1-3):101–124, 2001.

[HKPS04] Roland Heilmann, Daniel Keim, Christian Panse, and Mike Sips. Recmap: Rectangular map approximations. In *IEEE Symposium Information Visualization*, pages 33–40, 2004.

[IS06] Ryo Inoue and Eihan Shimizu. A new algorithm for continuous area cartogram construction with triangulation of regions and restriction on bearing changes of edges. *Cartography and Geographic Information Science*, 33(2):115–125, 2006.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Sixteenth annual ACM symposium on Theory of computing*, pages 302–311. ACM, 1984.

[KDW86] Ingrid Kretschmer, J Dörflinger, and F Warwick. Lexikon zur geschichte der kartographie.„enzyklopädie der kartographie", 1986.

[KNP04] Daniel A Keim, Stephen C North, and Christian Panse. Cartodraw: A fast algorithm for generating contiguous cartograms. *IEEE transactions on visualization and computer graphics*, 10(1):95–110, 2004.

[Koe36] Paul Koebe. *Kontaktprobleme der konformen Abbildung*. Hirzel, 1936.

[KR14] Linda Kleist and Benjamin Rahman. Unit contact representations of grid subgraphs with regular polytopes in 2d and 3d. In *International Symposium on Graph Drawing*, pages 137–148. Springer, 2014.

[McD19] Michael McDonald. Voter turnout data. *United States Elections Project*, Mar 2019.

[Mil08] Alisa Miller. Ted talk: The news about the news. "https://www.ted.com/talks/alisa_miller_shares_the_news_about_the_news#t-88454", 2008. Accessed on 06.07.2019.

[Mou07] David M Mount. Cmsc 754 computational geometry. *Lecture Notes for Spring*, 2007.

[NAK18] Sabrina Nusrat, Md Jawaherul Alam, and Stephen Kobourov. Evaluating cartogram effectiveness. *IEEE transactions on visualization and computer graphics*, 24(2):1077–1090, 2018.

[NK16] Sabrina Nusrat and Stephen Kobourov. The state of the art in cartograms. In *Computer Graphics Forum*, volume 35, pages 619–642. Wiley Online Library, 2016.

[NN19] Soeren Nickel and Martin Nöllenburg. Drawing k-linear metro maps. *arXiv preprint arXiv:1904.03039*, 2019.

[NSM+19] Soeren Nickel, Max Sondag, Wouter Meulemans, Markus Chimani, Stephen Kobourov, Jaakko Peltonen, and Martin Nöllenburg. Computing stable demers cartograms, 2019.

[PW+32] Charles Oscar Paullin, John Kirtland Wright, et al. Atlas of the historical geography of the united states. 1932.

[Rai34] Erwin Raisz. The rectangular statistical cartogram. *Geographical Review*, 24(3):292–296, 1934.

[Sch98] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[SSV18] Max Sondag, Bettina Speckmann, and Kevin Verbeek. Stable treemaps via local moves. *IEEE transactions on visualization and computer graphics*, 24(1):729–738, 2018.

[SvKF06] Bettina Speckmann, Marc van Kreveld, and Sander Florisson. A linear programming approach to rectangular cartograms. In *Progress in Spatial Data Handling*, pages 529–546. Springer, 2006.

[SW01] Ben Shneiderman and Martin Wattenberg. Ordered treemap layouts. In *IEEE Symposium on Information Visualization, 2001. INFOVIS 2001.*, pages 73–78. IEEE, 2001.

[Tim16] New York Times. Olympics - results. `"https://www.nytimes.com/interactive/2016/sports/olympics/rio-olympics-2016-medals-results.html"`, 2016. Accessed on 06.07.2019.

[Tob73] Waldo Tobler. A continuous transformation useful for districting. *Annals, NY Academy of Sciences*, 219:215 – 220, 1973.

[Tob04] Waldo Tobler. Thirty five years of computer cartograms. *ANNALS of the Association of American Geographers*, 94(1):58–73, 2004.

[Tor90] Jeffrey Scott Torguson. *Cartogram: a microcomputer program for the interactive construction of contiguous value-by-area cartograms*. PhD thesis, University of Georgia, 1990.

[U.S18] U.S. Census Bureau, Population Division. Annual estimates of the resident population, 2018.

[U.S19] U.S. Bureau of Economic Analysis. Gross domestic product (gdp) by state. https://bea.gov, Mar 2019.

[vKS05] Marc van Kreveld and Bettina Speckmann. Rectangular cartogram computation with sea regions. In *Proc. 22st Int. Cartographic Conference*, volume 10, 2005.

[vKS07] Marc van Kreveld and Bettina Speckmann. On rectangular cartograms. *Computational Geometry*, 37(3):175–187, 2007.

[Wat09] Social Watch. The social watch map 2005. "http://www.socialwatch.org/sites/default/files/statistics05/en/Mapa2005/ing-flash/index.html", 2009. Accessed on 06.07.2019.

[Wor01] Worldmapper. Map no. 205. "http://archive.worldmapper.org/display.php?selected=205", 2001. Accessed on 06.07.2019, direct link :'http://archive.worldmapper.org/posters/worldmapper_map205_ver5.pdf'.

[Wor18] Worldmapper. Ebola cases 1976-2016. "https://worldmapper.org/maps/ebola-cases-1976-2016/", 2018. Accessed on 06.07.2019.

[wPo29] 1929. Author of map: Joseph R. Grundy.