Improved Grid Map Layout by Point Set Matching

David Eppstein[*]
Dept. of Computer Science,
University of California,
Irvine

Marc van Kreveld[§]
Dept. of Information and
Computing Sciences,
Utrecht University

Bettina Speckmann[¶]
Dept. of Mathematics and
Computer Science,
TU Eindhoven

Frank Staals[§]
Dept. of Information and
Computing Sciences,
Utrecht University

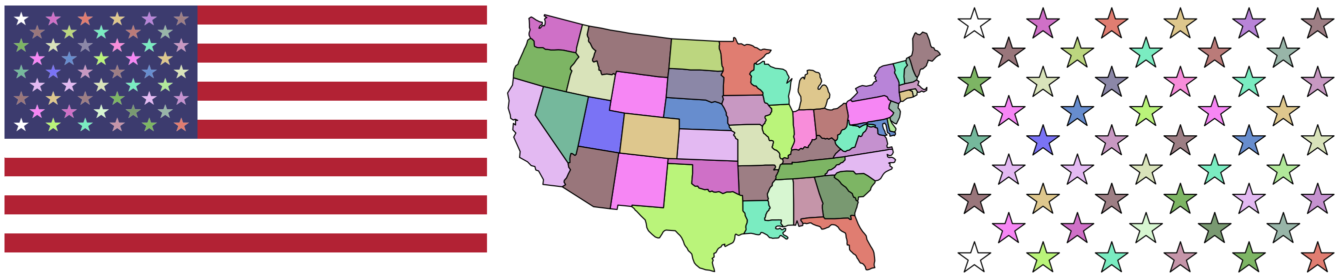
Figure 1: The states mapped onto the stars of the US Flag using the minimum L_1-distance matching under translation. The two white stars are reserved for Alaska and Hawaii.

Abstract

Associating the regions of a geographic subdivision with the cells of a grid is a basic operation that is used in various types of maps, like spatially ordered treemaps and OD maps. In these cases the regular shapes of the grid cells allows easy representation of extra information about the regions. The main challenge is to find an association that allows a user to find a region in the grid quickly. We call the representation of a set of regions as a grid a *grid map*.

We introduce a new approach to solve the association problem for grid maps by formulating it as a point set matching problem: Given two sets A (the centroids of the regions) and B (the grid centres) of n points in the plane, compute an optimal one-to-one matching between A and B. We identify three optimisation criteria that are important for grid map layout: maximise the number of adjacencies in the grid that are also adjacencies of the regions, minimise the sum of the distances between matched points, and maximise the number of pairs of points in A for which the matching preserves the *directional relation* (SW, NW, etc.). We consider matchings that minimise the L_1-distance (Manhattan-distance), the ranked L_1-distance, and the L_2^2-distance, since one can expect that minimising distances implicitly helps to fulfill the other criteria.

We present algorithms to compute such matchings and perform an experimental comparison that also includes a previous method to compute a grid map. The experiments show that our more global, matching-based algorithm outperforms previous, more local approaches with respect to all three optimisation criteria.

Index Terms: F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—Geometrical problems and computations

1 Introduction

Various types of maps associate the regions of a geographic subdivision to the cells of a grid. Prominent examples of such *grid maps* are spatially ordered treemaps [16] and OD-maps [17]. The regular shape of the grid cells allows easy representation of various types of information about the regions, see for example the London *BikeGrid* (`gicentre.org/bikegrid`) and Fig. 2. A major challenge when creating such grid maps is the layout of the regions, that is, the association of regions in the input subdivision to cells in the grid.

Tasks. We identify the tasks that grid maps should support when they are used. Firstly, a user needs to be able to *locate* the cell of a region in the grid map in order to retrieve the information it contains. Here we assume that the user is familiar with the rough layout of the geographic regions, otherwise she cannot do anything else than scanning all cells of the grid map. Secondly, a user may *compare* the information of two regions, after locating both cells. A user may also want to compare the information of a region with the surrounding regions. Thirdly, the user may *look for spatial patterns*, for example that the southern regions all have a relatively high value for some attribute X. Other possible tasks on grid maps exist; for a discussion on visual tasks we refer to [5].

For the location task, it is to be expected that a user will look for a region in the grid map based on her knowledge: if the user is looking for Louisiana on a grid map of the USA, she will look first in the bottom middle. If the cell there is for Alabama or Mississippi, then she may expect to find Louisiana a bit more to the left.

Optimisation criteria. The tasks and discussion above suggest that the following criteria are important to decide which regions of the map correspond to which cells of the grid map:

- Location (Louisiana should be in the bottom middle).
- Adjacency (Maine and New Hampshire should be adjacent).
- Relative orientation (Utah should be northwest of New Mexico).

While the spatial pattern task is not explicitly represented by these criteria, it is likely that the location criterion will help to fulfill it. With these criteria in mind we formalize the problem of assigning the regions to the grid cells.

Formal problem definition. We model the association problem for grid maps as a weighted point set matching problem between the centroids of the input regions and the centres of the grid cells.

[*]e-mail: eppstein@ics.uci.edu
[§]e-mail: {m.j.vankreveld, f.staals}@uu.nl
[¶]e-mail: speckman@win.tue.nl

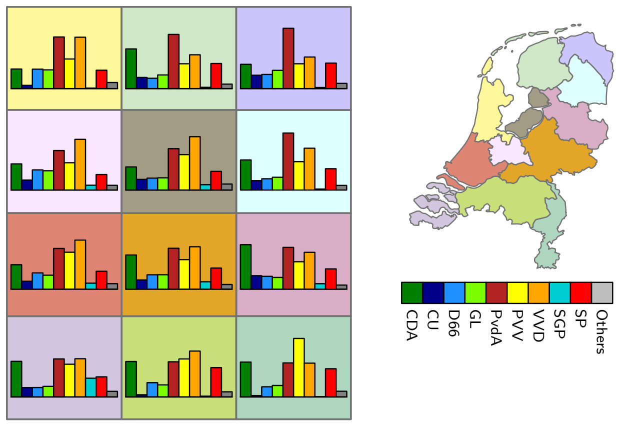

Figure 2: Election results from the Netherlands in a grid map.

Let $\mathcal{M}$ be a map with n regions and let $\mathcal{G}$ be a grid with n grid cells. To define the distance between regions and grid cells we will use the centroids of the regions and the centroids (middle) of the grid cells. Note that we need to bring the geographic map and the grid map into the same coordinate system before distance has any meaning. First of all, we need to choose a map projection to obtain two planar point sets. Note that the choice of the map projection influences the result of the matching. Second, we translate one point set with respect to the other and scale by x- and y-coordinates. Rotation and other transformations are less important, although there are cases where a rotated grid can represent a map better than a standard grid (for example, Japan or Malaysia).

Let A be the set of centroids of the regions in $\mathcal{M}$ and let B be the set of grid cell centres of $\mathcal{G}$. We want to compute a one-to-one matching ϕ between the regions in the map (or points in A) and the cells in the grid (or points in B), such that the resulting grid map is as similar to $\mathcal{M}$ as possible. The three criteria we listed before for the quality of the matching ϕ are now: (*i*) minimising the sum of the distances between matched points—one from A and one from B—under translation, scaling, and both translation and scaling of point set A, (*ii*) maximising the number of adjacencies in $\mathcal{M}$ that the matching preserves in $\mathcal{G}$, and (*iii*) maximising the number of pairs of points in A for which the matching preserves the *directional relation*. That is, if a point a_2 lies northwest of point a_1, then we would like $\phi(a_2)$ to lie northwest of $\phi(a_1)$ as well.

Related work. We distinguish research that presents related visualisations and research that discusses related algorithmic ideas, in particular point set matching problems.

There is a longstanding cartographic tradition of associating statistical information with the regions of a map. Such information can be captured in various forms of visualizations, such as graphs or charts. Often these visualizations are directly overlayed onto the associated base map, for example in *proportional symbol maps* [10]. Such an overlay necessarily has to deal with occlusions, both of the symbols themselves and of the base map. The work of Cabello et al. [4] tries to alleviate some of the issues arising. Also van Kreveld et al. study problems related to placing diagrams on maps [13]. In particular, they try to minimize the amount of overlap between diagrams of different regions, and between the diagrams and other regions and features of the map. Another approach are the *necklace maps* [11] introduced by Speckmann and Verbeek which move visualization to a necklace surrounding the map. Still, none of these methods can fully avoid visual clutter when detailed statistical visualizations are combined with small geographic regions. If the exact geographic location of the regions and their associated visualizations is not paramount, grid maps are an attractive alternative which provides equal space for the visualizations of every region.

Grid maps are a simple type of spatially ordered treemap [16]. A spatially ordered treemap, or spatial treemap for short, need not have the rigorous grid structure of a grid map. A (spatial) treemap can fill a rectangular region with any number of rectangles, each of which may have a different size. Thus, a grid map can be seen as a single-level spatial treemap in which all rectangles have the same size and orientation, and are nicely aligned in a regular grid. We focus on grids because they are the simplest partition of a space, where all cells get the same space to show extra information. These properties may improve the readability of the visualization. Note, though, that our algorithms do not require the rectangular grid structure to function properly, see for example Fig. 1, which uses a different type of grid.

Wood et al. [18] use a grid map, or a *spatial matrix* as they call it, to visualize information from the London BikeGrid: London's bicycle hiring scheme. They use an adapted spatial treemap [16] to map the docking-stations to the grid cells. Each cell itself displays the number of available bikes (in the corresponding docking-station) using a graph and different colours.

Grid maps are also related to OD maps [17, 18, 9]. OD maps are used to display the flow in Origin-Destination data. An OD map is a grid in which each cell corresponds to an origin region. Each cell again contains a grid in which each destination region is represented by a cell. When the grids preserve the spatial layout of the underlying map they can show spatial patterns in the data, an improvement over OD matrices introduced by Voorhees [15]. Hence these OD maps can be seen as two-level grid maps. A grid map is therefore not a new type of map, but the name for a grid whose cells are associated with geographic regions or locations in an optimised manner.

We listed three optimisation criteria for grid map layout. They all concern a sum of counts or distances, and therefore the global approach of *matching* is natural. The first optimisation criterion relates to minimising the distance when matching two point sets. Many papers on this topic exist in computational geometry and shape matching. For surveys, see Alt and Guibas [1] and Veltkamp and Hagedoorn [14]. Existing results differ in the matching distance used, whether the matching is bipartite or not, whether the point sets have the same cardinality or not, and which transformations can be applied to the one point set to match it best with the other. Distances of matchings can be based on sums of matched points in the L_1 (Manhattan), L_2 (Euclidean), or L_2^2 metric, but they can also be based on minimising the maximum distance of the matched points. The latter type of matchings are called bottleneck matchings. Another well-known metric is the Earth mover's distance, which is useful when the points have different weights. Transformations that may be applied before matching can be translation, scaling, rotation, and reflection. Given our objectives, rotation and reflection are less suitable transformations. Translation and scaling on x- and y-coordinates, however, are suitable to place a geographic map into the same coordinate system as a grid map.

Efrat et al. [7] present an $O(n^5 \log^2 n)$ algorithm to compute a minimum bottleneck matching of two sets of n points under translation. A bottleneck matching minimizes the maximum distance between a pair of matched points, However, such a matching has the unfortunate effect that all point pairs with smaller distances can be matched arbitrarily without influencing the resulting distance. Therefore, we study matchings that minimize the sum of the distances between matched pairs of points.

The second optimisation criterion is preserving the maximum number of adjacencies. Unfortunately, even a simple version of the problem is NP-hard [3]. Furthermore, it is clear that concentrating purely on adjacencies will not give good grid map layouts because the other two criteria will be grossly violated in many cases. Hence we will not optimise this criterion explicitly.

The third optimisation criterion is preserving directional relations. To our knowledge, this type of criterion has not been studied before in point set matching. The computational complexity of the problem is unknown.

It seems likely that using a matching that minimises sums of distances will also be reasonably good for preserving adjacencies and directional relations. Hence we concentrate on this idea, and later analyse the performance of all methods on these criteria experimentally. The metrics we consider are the L_1-distance, the L_2^2-distance, and a ranked L_1-distance. The ranked L_1-distance can compensate for situations where the regions on a map are unevenly spaced in x- or y-direction, like the USA: horizontal spacing for the western States is considerably larger than for the eastern States. The ranked L_1-distance will ignore this aspect in its value. Furthermore, this distance allows us to prove a theoretical approximation bound for optimising the number of correct direction relations.

We do not consider bottleneck matchings for the reasons we mentioned earlier. We also do not consider the Earth mover's distance because we do not have weighted points. Finally, we do not consider the sum of L_2-distances because even for a given matching, we cannot compute the optimal translation analytically, implying that computing the optimal matching using the sum of L_2-distances is not possible.

Results and organisation. We present algorithms to solve the association problem for grid maps based on optimal matchings between two point sets. In the next section we show how to compute such optimal matchings. We study minimum distance matchings using the L_1-distance under translation, scaling, and the combination of translation and scaling, and show that these can be solved in polynomial time. Whereas a matching itself can be computed efficiently, optimising matchings over translations and scalings will not be feasible for large-size instances, however. We also study a ranked L_1-distance, since it gives theoretical approximation guarantees for optimising the number of relative orientations. The proofs for the approximation results are given in a forthcoming extended version of the paper. We also discuss our implementation using an LP solver and show that it can easily incorporate cases where not all grid cells should be used.

Section 3 presents an experimental analysis of matchings using the L_1-distance, L_1-distance by ranks, and L_2^2-distance. The latter algorithm is by Cohen and Guibas [6], who solve minimum L_2^2-distance matching under translations only. We compare matching by these distance measures with the approach used by Wood and Dykes [16] and report the summed distances, the percentage of adjacencies kept, and the percentage of correct relative orientations. Our data consists of the maps of France, the United States, and the London boroughs. Our matching approach outperforms previous approaches to solve the association problem on the three criteria.

The matching approach does not depend on the grid pattern, we can apply it to other patterns as well. For example, we can associate the States in the US to the stars of the flag, as shown in Fig. 1.

2 Computing a Grid Map

2.1 Minimising Distance

We first introduce some notation. For a point $a = (a_x, a_y)$ and a translation $t = (t_x, t_y)$ we write $a + t = (a_x + t_x, a_y + t_y)$. We also use this notation for a set of points: $A + t = \{a + t \mid a \in A\}$. Similarly, for a scaling $\lambda = (\lambda_x, \lambda_y)$ we write $\lambda a = (\lambda_x \cdot a_x, \lambda_y \cdot a_y)$. A transformation (either translation or scaling) in which both components have the same value c we denote by $\overline{c} = (c, c)$.

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

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

where $d(a, b)$ denotes the L_1-distance (Manhattan-distance) between a and b. Additionally, we define $D_{\mathcal{T}}(\phi, t) = D(\phi, t, \overline{1})$, $D_\Lambda(\phi, \lambda) = D(\phi, \overline{0}, \lambda)$ and $D_I(\phi) = D(\phi, \overline{0}, \overline{1})$.

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

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

Since we are using the L_1-distance we can decompose d into a horizontal and a vertical component: $d(a, b) = x(a, b) + y(a, b)$ with $x(a, b) = |a_x - b_x|$ and $y(a, b) = |a_y - b_y|$. We generalise this notion to D, which gives us $D(\phi, t, \lambda) = X(\phi, t, \lambda) + Y(\phi, t, \lambda)$. The functions $X_{\mathcal{T}}, Y_{\mathcal{T}}, X_\Lambda$, etc. are defined accordingly.

Minimising L_1. The easiest case one can consider is to compute a matching that minimises D_I: a minimum distance matching without translation or scaling. This problem can be solved using Vaidya's method [12] in $O(n^2 \log^3 n)$ time.

Minimising L_1 under translation. To find a minimum distance matching under translation, i.e. a matching that minimises $D_{\mathcal{T}}$, we identify a (finite) set of translations $T \subset \mathcal{T}$ that contains an optimal translation. We then use Vaidya's method for each translation in T to compute an optimal matching.

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

We now observe that for any matching ϕ between point sets A and B that are not x-aligned there is a horizontal translation that does not increase $X_{\mathcal{T}}(\phi)$ (in most cases x-aligning the point sets will even decrease the distance $X_{\mathcal{T}}(\phi)$). See Fig. 3 for an illustration. Hence:

Lemma 1. *Let A and B be two non x-aligned sets of n points in the plane, and let ϕ be any one-to-one matching between A and B. Then there is a horizontal translation $t^* \neq \overline{0}$ such that $A^* = A + t^*$ and B are x-aligned and $D_{\mathcal{T}}(\phi, t^*) \leq D_I(\phi)$.*

Proof. We consider only horizontal translations t^* so it follows that $Y_{\mathcal{T}}(\phi, t^*) = Y_I(\phi)$. The function $X'(t) = X_{\mathcal{T}}(\phi, t)$ is piecewise-linear in t and has its minimum at a breakpoint, say t'. Since X' is the sum of a set $\mathcal{F} = \{f_a \mid f_a(t) = |a_x + t_x - \phi(a)_x| \wedge a \in A\}$ of piecewise-linear functions it follows that there is a function $f_a \in \mathcal{F}$ which also has its minimum at t'. The minimum value of f_a is zero and occurs at its breakpoint. Hence $f_a(t') = 0$. This means that $a + t'$ x-aligns with $\phi(a)$. We conclude that there is a translation $t^* = t' \neq \overline{0}$, that x-aligns A^* and B, and minimises X'. The lemma follows. $\square$

Analogously there is a vertical translation that y-aligns the two sets of points and does not increase $Y_{\mathcal{T}}$.

Consider the set T of translations that both x-align and y-align A and B. A translation $t \in T$ x-aligns a pair of points (a, b), and independently y-aligns a pair of points (a', b'). This means T contains at most n^4 translations.

Now if ϕ and t are a matching and a translation that minimise $D_{\mathcal{T}}$ it follows from Lemma 1 (and its counterpart for y-aligning the point sets) that we can x- and y-align the point sets without increasing the total distance of matching ϕ. Hence, T contains an optimal translation t^*. This means we can find an optimal matching ϕ^* by computing a minimum distance matching for all translations in T. By using the algorithm of Vaidya [12] to compute the point set matchings we obtain the following result:

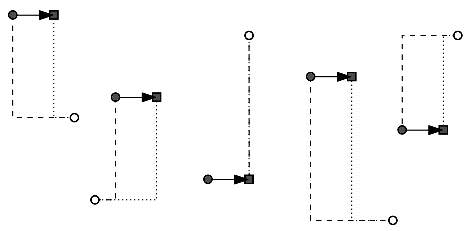
Figure 3: We can improve a matching between A (grey) and B (white) indicated by the dashed lines by x-aligning the point sets (the dotted lines).

Theorem 1. *Given two sets A and B of n points in the plane, a one-to-one matching ϕ^* and a translation t^* that minimise $D_\mathcal{T}$ can be computed in $O(n^6 \log^3 n)$ time.*

The main difficulty in improving this result is that $X^*(t) = X(\phi^*_t, t)$, where ϕ^*_t denotes an optimal matching for horizontal translation t, is not unimodal. Therefore X^* may have several local minima, which means we cannot use a binary search to find an optimal translation t^*. Instead, we have to compute a matching for all translations in T.

Minimising L_1 under scaling. For scaling we can use the same procedure as for translation: we prove that there is an optimal scaling that x-aligns and y-aligns A and B and does not increase the total distance. We again have a set of at most n^4 scalings that is guaranteed to contain an optimal scaling. Hence:

Theorem 2. *Given two sets A and B of n points in the plane, a one-to-one matching ϕ^*, and a scaling λ^* that minimise D_Λ can be computed in $O(n^6 \log^3 n)$ time.*

Minimising L_1 under both translation and scaling. We can use same the approach, but now we x-align (y-align) two distinct pairs of points. We obtain:

Theorem 3. *Given two sets A and B of n points in the plane, a one-to-one matching ϕ^*, a translation t^*, and a scaling λ^* that minimise D can be computed in $O(n^{10} \log^3 n)$ time.*

Proof. Analogous to the proof of Lemma 1 we can show that $X'(t, \lambda) = X(\phi, t, \lambda)$ has its minimum at a breakpoint $(\hat{t}, \hat{\lambda})$. This again corresponds to x-aligning a point $\hat{a}$ with $\phi(\hat{a})$. What remains to show is that there is a second point a' that we can x-align with $\phi(a')$. It is easy to see that if there is only one pair of points x-aligned, say $\hat{a}$ and $\phi(\hat{a})$, there are a translation t' and scaling λ' that will keep $\hat{a}$ x-aligned with $\phi(\hat{a})$ and minimise the total distance between $\{\hat{\lambda}a + \hat{t} \mid a \in A \setminus \{\hat{a}\}\}$ and $B \setminus \{\phi(\hat{a})\}$. Using the same argument as before it follows that this distance is smallest when we x-align a point a' with $\phi(a')$. We conclude that there is a translation t^*, namely the combination of $\hat{t}$ and t', and a scaling, namely the combination of $\hat{\lambda}$ and λ', that minimises X' and x-aligns two pairs of points. This completes the proof. □

Using that B is a grid. The above results all hold for arbitrary sets of points A and B in the plane. However, for our grid maps we can use that the points in B are grid points of a regular grid. Any two points b_1, b_2 in the same column of the grid have the same x-coordinate. So x-aligning a point $a \in A$ with b_1 has the same effect as x-aligning a with b_2. The same holds for any two points in the same row. Hence, we can improve the running time of our algorithms slightly:

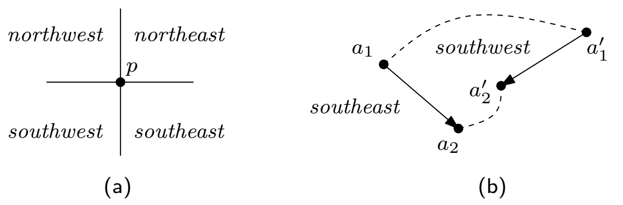

Figure 4: (a) The areas in the plane corresponding to each direction. (b) The directional relation between a_1 and a_2 is not preserved, (a_1, a_2) is an x-inversion.

Corollary 1. *Given a set A of n points in the plane and a set B of n grid points on an $R \times C$ grid of size n, a minimum L_1-distance matching under translation or scaling can be computed in $O(nCnR \cdot n^2 \log^3 n) = O(n^5 \log^3 n)$ time. A minimum L_1-distance matching under translation and scaling can be computed in $O(n^8 \log^3 n)$ time.*

2.2 Preserving Directional Relations

The third criterion that we consider is preserving directional relations. Let A and B be two sets of n points in which no two points have the same x- or y-coordinate (note that this is not the case when B is a grid), and let $dir(p, q)$ denote the directional relation of q with respect to p (see Fig. 4 (a)). The goal is now to find a matching $\phi^* \colon A \to B$ that maximises the number of pairs $(a_1, a_2) \in A \times A$ for which $dir(a_1, a_2) = dir(\phi^*(a_1), \phi^*(a_2))$. Stated differently, we are looking for a matching ϕ^* that minimises the number of *out-of-order* pairs W defined as

$$W(\phi) = |\{(a_1, a_2) \mid (a_1, a_2) \in A \times A \wedge dir(a_1, a_2) \neq dir(\phi(a_1), \phi(a_2))\}|.$$

To avoid many nested brackets we will write $a' = \phi(a)$ from now on. Furthermore, we observe that translations and scalings do not influence W.

A 4-approximation algorithm for minimising W. We now describe an algorithm to compute a matching that approximately minimises W. Let $x\text{-}rank_P(p)$ denote the *x-rank* of point $p \in P$, that is, the number of points in P to the left of p. For points $p \in P$ and $q \in Q$ we write $p \prec_x q$ for $x\text{-}rank_P(p) < x\text{-}rank_Q(q)$. The y-rank and $\prec_y$ are defined analogously.

For a given matching, $(a_1, a_2) \in A \times A$ is an *x-inversion* if (and only if) $a_1 \prec_x a_2$ and $a'_1 \succ_x a'_2$, or $a_2 \prec_x a_1$ and $a'_2 \succ_x a'_1$. See Fig. 4 (b). Similarly we define a *y-inversion*. An *inversion* is an x-inversion, a y-inversion or both. We denote the number of x-inversions and the number of y-inversions of matching ϕ by $I_x(\phi)$ and $I_y(\phi)$, respectively. It is easy to see that there is a one-to-one correspondence between the number of out-of-order pairs $W(\phi)$ of matching ϕ and the number of inversions $I(\phi)$, i.e. $W(\phi) = I(\phi)$.

We define a distance measure w between points $a \in A$ and $b \in B$:

$$w(a, b) = |x\text{-}rank_A(a) - x\text{-}rank_B(b)| + |y\text{-}rank_A(a) - y\text{-}rank_B(b)|.$$

We now compute a minimum distance matching ϕ with w as distance measure. The distance measure w is simply the L_1-distance on the ranks of the points, which means we can use Vaidya's algorithm [12] to compute ϕ. Let $D(\phi)$ again denote the total distance of matching ϕ, then we can prove that the matching that minimizes D is a 4-approximation algorithm for minimizing W:

Theorem 4. *Given two sets A and B of n points in the plane, we can compute a one-to-one matching ϕ where $W(\phi) \leq 4 \cdot \min_{\phi^* \in \Phi} W(\phi^*)$ in $O(n^2 \log^3 n)$ time.*

The proof of this theorem can be found in a forthcoming extended version of the paper.

2.3 Implementation

We implemented a tool that computes an $R \times C$ grid map $\mathcal{G}$ of a given input map $\mathcal{M}$ for a specified number R of rows and C of columns. The tool itself is implemented in Scala, and uses lpsolve [2] to solve the underlying point set matching problems. The global approach is as follows.

We start by constructing an (empty) $R \times C$ grid, in which the grid cells have heights and widths such that $\mathcal{G}$ and $\mathcal{M}$ have the same (size) bounding box. For each of the regions and each of the grid cells we compute its centroid, thus obtaining the sets of points A and B. We then generate all horizontal transformations (translations or scalings): one for each pair consisting of a point in A and a column in the grid. Analogously, we generate all vertical transformations. By combining the horizontal and vertical transformations we obtain a set T of $O(n^3)$ transformations. For each of these transformations in turn, we apply the transformation on A, and compute a minimum distance matching between the resulting set and B. We pick the matching that minimizes the distance over all transformations in T, and use it to map each region in $\mathcal{M}$ onto a cell in $\mathcal{G}$.

The point set matchings can be solved using linear programming, in particular by using Vaidya's algorithm [12]. However, for our implementation we use the following simpler, but slower, LP-formulation, which we solve using lpsolve [2].

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

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

subject to:

$$\sum_{b \in B} f_{ab} = 1 \qquad \forall a \in A$$

$$\sum_{a \in A} f_{ab} = 1 \qquad \forall b \in B$$

$$0 \leq f_{ab} \leq 1 \qquad \forall a \in A, b \in B$$

Since all supplies and demands have integer values it can be shown that all variables in the optimal flow f_{ab} also have integer values [8]. This means that f_{ab} represents a one-to-one matching $\phi \in \Phi$ that matches a to b if and only if $f_{ab} = 1$. Thus, the objective function expresses the total distance $D_I(\phi)$. It follows that the matching computed by this linear program minimises D_I.

In case matching the points a_1 to b_1 and a_2 to b_2 yields the same L_1-distance as matching a_1 to b_2 and a_2 to b_1 (note that this is not a degenerate case), we make sure our tool chooses the matching that minimises the maximum distance.

If we wish to match A to a set B with $m > n$ points, that is, we allow empty grid cells, we can relax the second constraint to $\sum_{a \in A} f_{ab} \leq 1$.

3 Evaluation

In this section we give an experimental evaluation of our methods. We compare the results from the different distance based methods to each other, and to a method based on spatial treemaps by Wood and Dykes [16]. To determine the quality of the resulting grid maps we measure the distance between the point sets after translation, the number and percentage of preserved adjacencies, and the number and percentage of preserved directional relations. When we count the number of directional relations we consider points mapped to the same row (column) to have the correct north-south (east-west) relationship. Two regions or grid cells are adjacent if the intersection of their closed boundaries is nonempty. In particular this means each grid cell has at most eight neighbours.

Additionally, we use a qualitative analysis based on the colouring of the regions. Similar to Wood and Dykes [16], we map a CIE L*a*b* colour space with $L^* = 50$ onto the input map. Each region is assigned the colour of its centroid. The same colour is used for the grid cell corresponding to this region. The idea is that in a good grid map the colour changes gradually, as it does in the input map. This indicates the relative positions in the grid are similar to those in the input map.

The matchings that we consider in our evaluation are: (*i*) a matching that minimises the L_1-distance without translation or scaling (i.e. D_I), (*ii*) a matching that minimises the L_1-distance under translation, (i.e. $D_\mathcal{T}$), (*iii*) a matching that approximates the minimum number of out of order pairs W, and (*iv*) a matching that minimises the total L_2^2-distance under translation. This last matching is due to Cohen and Guibas [6]. Minimising the L_1-distance under (only) scaling yields similar results to minimising under (only) translation, so we do not show the results here. Computing a minimum distance matching under both translation and scaling was computationally unfeasible. In the remainder of this section we refer to the last three methods by L_1, W, and L_2^2. The method that minimises D_I is simply called I.

We compare results of the distance based methods with a modified version of Wood and Dykes's spatial treemaps [16]. The modifications make sure the result is a grid map, rather than an arbitrary spatial treemap. The algorithm recursively processes the cells incident to a shortest side of the grid. That is, it processes a single row or column of the grid. For each of these cells in turn it finds the point $a \in A$ that is closest to its centroid $b \in B$. This pair is added to the matching. The remaining grid cells again form grid, one that is exactly one row or column smaller than before, which is processed recursively. We refer to this method as *SpatialGrid*.

We focus our evaluation on the quality of the resulting grid maps, rather than on the running time of the algorithms. For the maps presented here the entire tool takes only a few seconds when using the SpatialGrid, I, W, or L_2^2 method. So computing a single minimum distance matching takes roughly the same amount of time as the greedy algorithm used in the SpatialGrid method. The L_1 method however requires the computation of very many minimum distance matchings. Therefore it is significantly slower than the other methods. For the larger maps the L_1 method already takes several hours.

United States. We use our algorithms to construct a grid map of the United States. To prevent artificially inflating the bounding box of the map we consider only the 48 contiguous states. Fig. 5 shows the resulting grid map for each of the methods, and Table 1 contains the measurements. We can see that the L_1 and L_2^2 methods minimise their respective distances. The I, SpatialGrid, and W methods have a much larger total distance. What is perhaps somewhat surprising is that the I method has a smaller L_1-distance than the SpatialGrid method, but if we use the L_2^2-distance then SpatialGrid has a smaller total distance. Most likely, this is since the L_2^2-distance is more sensitive to large differences in a single component of the distance. Consider for example Florida (FL). In the I method the x-component of the distance between the Florida and its grid cell

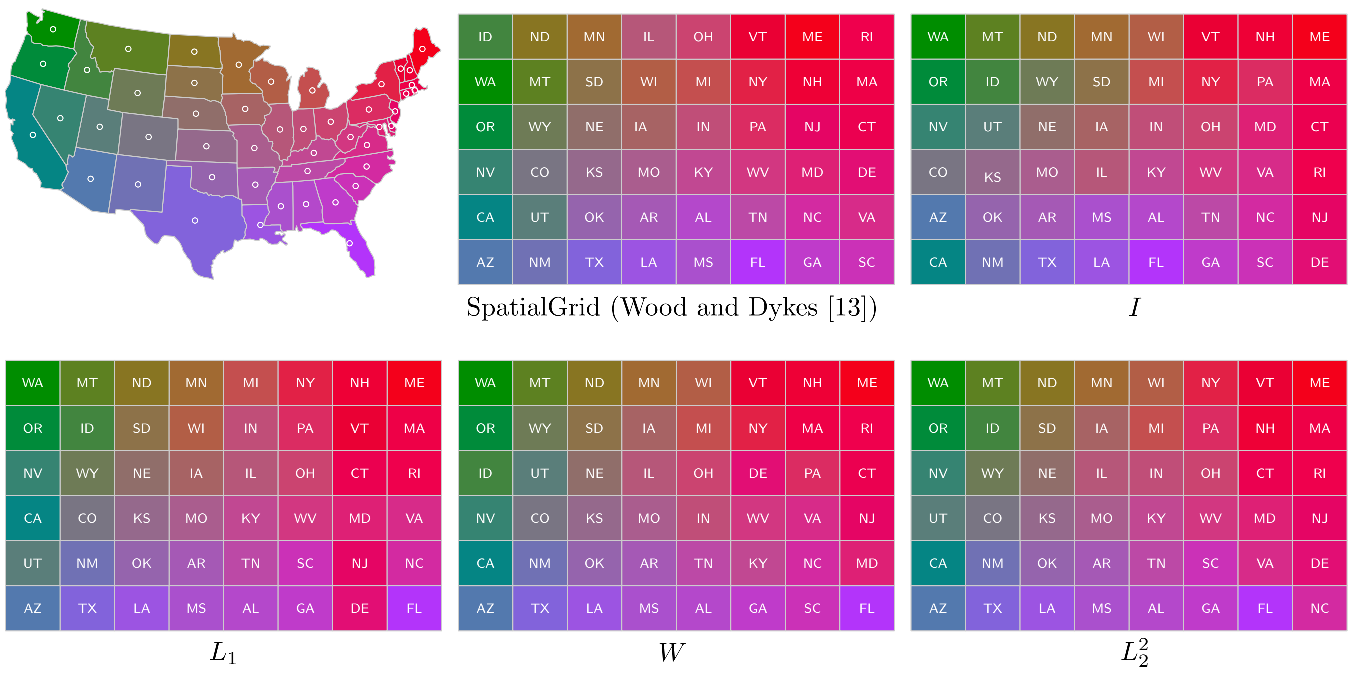

Figure 5: The results of the three methods on the contiguous states of the US.

Method	Distance			Directional Rel.		Adjacencies	
	L_1	L_2	L_2^2	#	%	#	%
SpatialGrid (Wood and Dykes [16])	4545	3592	300482	2008	89.01%	77	73.33%
I	4035	3342	311327	2024	89.72%	79	75.24%
L_1	2838	2355	166060	2046	90.69%	78	74.29%
W	4221	3352	273273	2098	93.00%	79	75.24%
L_2^2	2929	2260	139110	2096	92.91%	83	79.05%

Table 1: United States.

is quite large (i.e. larger than in the grid map corresponding to SpatialGrid). This has a much larger influence on the L_2^2-distance than on the L_1-distance.

The 4-approximation algorithm for minimising W preserves most directional relations: 93% of the pairs of points in A have the correct directional relation. The W method is closely followed by the L_2^2 method. The method that preserves the least directional relations is SpatialGrid. This is still 89%. Almost all methods preserve around 74% of the adjacencies. The L_2^2 method performs a bit better here, preserving 79% of the adjacencies.

The grid maps in Fig. 5 confirm the quantitative results. The grid map corresponding to the SpatialGrid method has a few places where cells seem out of place. In particular, the group of cells IL, OH, WI, MI (in the middle of the two topmost rows). The grid maps for the I and L_1 methods also show some out of order cells. For the I method most notably the cells corresponding to CO, and CA. In the L_1 grid map those corresponding to NJ and DE. The grid maps for the W and L_2^2 methods show the most natural changes of colours, indicating the least distortions.

France. We also use our methods on 96 departments of France. The resulting grid maps are shown in Fig. 6. The quantitative results can be found in Table 2. For the total distance of the matchings we see similar results to those of the US. The L_1 and L_2^2 methods have the smallest distances, the distance of the W method is somewhat similar to that of I, and SpatialGrid has the largest distance. We again see that the W and L_2^2 methods preserve most directional relations. In this case almost all pairs of points have their correct directional relation: around 97%. The L_1 and I methods also perform very well, preserving roughly 95% of the correct directional relations. SpatialGrid performs significantly worse: preserving only 88% of the directional relations. If we look at the grid map corresponding to SpatialGrid in Fig. 6 we can see three cells in the top right corner which should have been much further to the southwest. This problem is due to the greedy matching scheme. The point (region) $a \in A$ matched to such a cell was never the point closest to any other cell, and hence when we are processing these last cells these points a are the only points left that we can match to.

In terms of the number of preserved adjacencies we can also see that SpatialGrid does not perform well. The same applies for the I method. Both preserve only 69% of the adjacencies, whereas the L_1 and L_2^2 methods manage to preserve 76% and 80% of the adjacencies, respectively. The W method even manages to preserve 83% of the adjacencies.

The grid map with the smoothest color changes is again the one corresponding to the L_2^2 method. Both the I and L_1 methods show a natural change in color as well. The grid map corresponding to W shows an out of place green cell in the bottom-left, and places the two most purple cells corresponding to Corsica a bit further to the west than we would expect them.

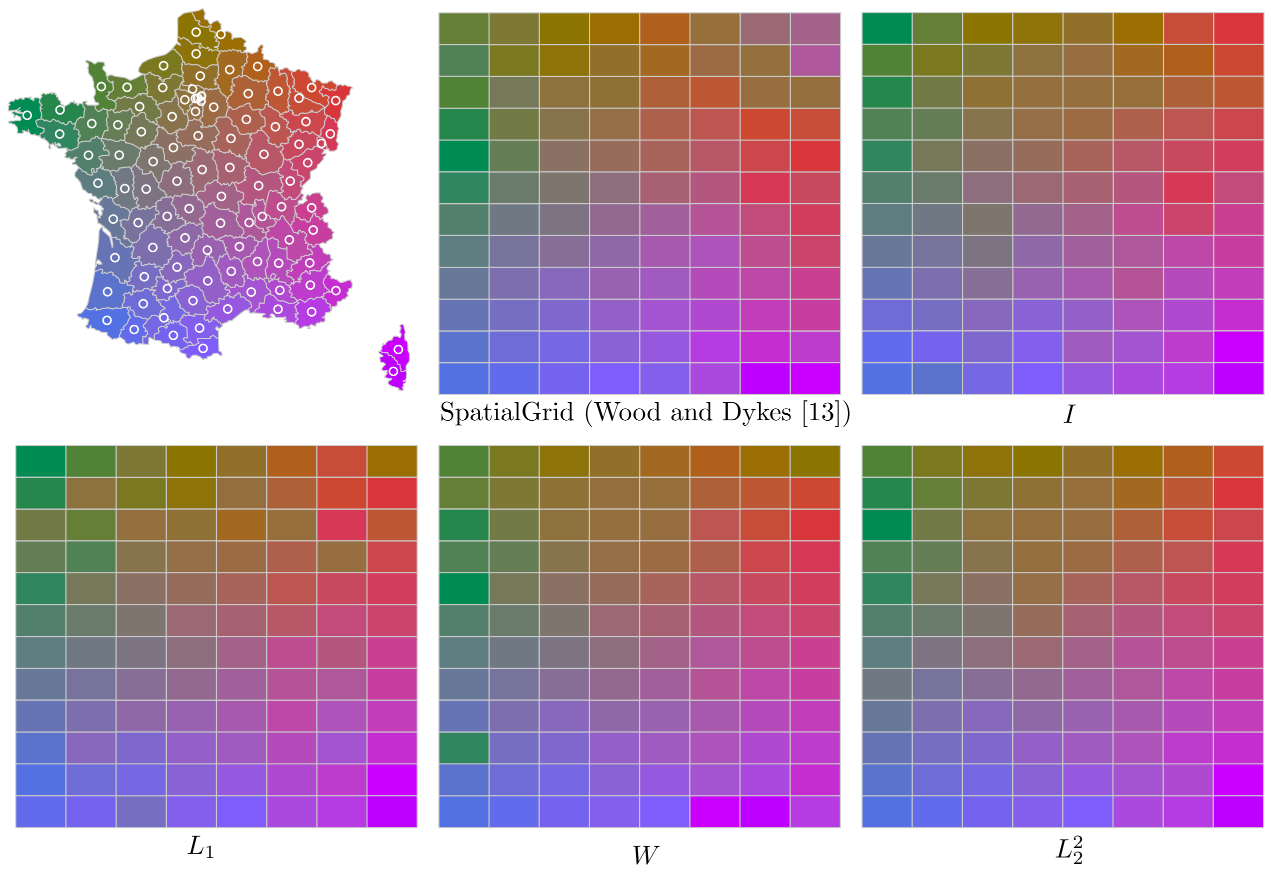

Figure 6: The results of the three methods on the 96 departments in France.

Method	Distance			Directional Rel.		Adjacencies	
	L_1	L_2	L_2^2	#	%	#	%
SpatialGrid (Wood and Dykes [16])	10666	8094	1066752	8040	88.16%	166	69.46%
I	9161	7462	698697	8620	94.52%	165	69.04%
L_1	8622	7086	654457	8718	95.59%	183	76.57%
W	9359	7167	647438	8874	97.30%	197	82.43%
L_2^2	8875	6889	595280	8894	97.52%	193	80.75%

Table 2: France.

London boroughs. Finally, we compute grid maps for the 33 London boroughs. We use a 6×6 grid in which we manually specified which cells (not) to use. Recall that the SpatialGrid method is an adaptation of a method of Wood and Dykes to produce a spatial treemap, and that spatial tree map contains no empty spaces. There are many possible ways to incorporate empty cells in our adaptation of the algorithm. However, none of these ways is obviously the right one, and each option to deal with empty cells gives different results, making the choice rather arbitrary. Therefore, we only compare the results of the four matching-based methods. The results can be found in Table 3 and Fig. 7. The results are somewhat similar to those of the United States and France. In terms of preserved directional relations and adjacencies the L_1 method performs a bit worse than before. However, if we look at the colours of the grid maps in Fig. 7 the grid map for the L_1-method still shows a very natural gradient. The W method seems to perform best on this input map. Both the W method as the L_2^2 method preserve the most directional relations and adjacencies, but the grid map for W has a slight edge over those of the I and L_2^2 methods. This is mainly because of the positioning of the orange and dark-red cells in the upper right corner of the grid maps.

4 Concluding Remarks

We studied grid maps: a schematic representation of the regions of a normal map where every region corresponds one-to-one to a grid cell. To compute the correspondence between the regions and grid cells we investigated point set matching problems between two sets A and B of n points in the plane. To obtain a good grid map we considered three optimisation criteria. One of these criteria, maximising the number of adjacencies that the matching preserves, is NP-hard. For one of the other criteria, minimising the sum of L_1-distances between matched points under translation and/or scaling, we gave polynomial-time algorithms. For the last criterion, maximising the number of pairs of points in A for which the matching preserves the directional relation, we gave a 4-approximation algorithm. We implemented our methods and evaluated them on several

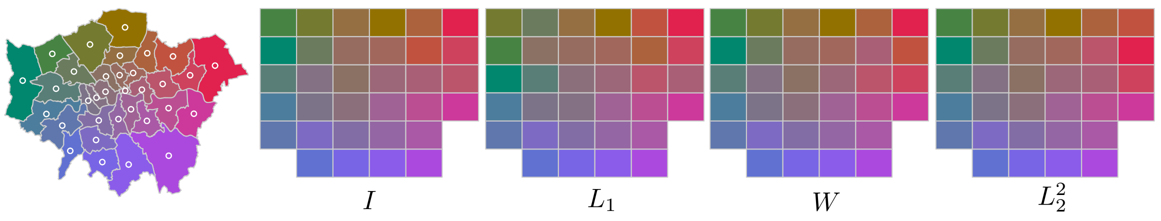

Figure 7: The results of the three methods on the London boroughs.

Method	Distance			Directional Rel.		Adjacencies	
	L_1	L_2	L_2^2	#	%	#	%
I	2897	2296	182257	1040	98.48%	59	72.84%
L_1	2803	2286	200593	1008	95.45%	54	66.67%
W	2936	2277	177927	1042	98.67%	61	75.31%
L_2^2	2890	2228	172089	1042	98.67%	61	75.31%

Table 3: London boroughs.

maps. The results of the three methods are comparable and generally good. In all considered maps our distance-based methods produce better grid maps than previous methods. Our experiments also confirm our hypothesis that the methods using a minimum distance matching will preserve many directional relations and adjacencies.

In our setting the grid with the correct number of cells is given as part of the input. An interesting problem that remains is to determine which grid cells of a slightly too large grid to use. If the matching algorithm makes the decision, we may have unused grid cells in the middle. For most of our maps the minimum L_2^2-distance matching under translation gives (one of) the best results. Hence it would be interesting to try to minimise the L_2^2-distance under scaling as well. Another open problem is related to preserving the directional relations. The complexity of optimising this criterion is not known, and it may be possible to obtain a better approximation factor or a polynomial-time approximation scheme.

Acknowledgements

The authors are grateful to Jason Dykes and Jo Wood for their comments on an earlier version of this paper and for providing insight in their spatial treemap implementation. This research was supported in part by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.022.707 and 612.001.022, the National Science Foundation under grant 0830403, and by the Office of Naval Research under MURI grant N00014-08-1-1015.

References

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

[2] M. Berkelaar, K. Eikland, P. Notebaert, et al. lpsolve: Open source (mixed-integer) linear programming system, 2010. URL `http://lpsolve.sourceforge.net/`.

[3] F.-J. Brandenburg. On the complexity of optimal drawings of graphs. In *WG*, volume 411 of *Lecture Notes in Computer Science*, pages 166–180. Springer, 1989.

[4] S. Cabello, H. Haverkort, M. van Kreveld, and B. Speckmann. Algorithmic aspects of proportional symbol maps. *Algorithmica*, 58(3):543–565, 2010.

[5] W. Cleveland. *The Elements of Graphing Data*. AT&T Bell Laboratories, 1994.

[6] S. Cohen and L. Guibas. The earth mover's distance under transformation sets. In *Proc. 7th IEEE International Conference on Computer Vision*, volume 2, pages 1076 –1083, 1999.

[7] A. Efrat, A. Itai, and M. J. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31:1–28, 2001.

[8] F. Hillier and G. Lieberman. *Introduction to Mathematical Programming*. McGraw-Hill, 1990.

[9] A. Slingsby, M. Kelly, J. Wood, and J. Dykes. OD maps for studying historical internal migration in Ireland. In *Proc. IEEE Conference on Information Visualization*, pages 239–251, 2011.

[10] T. Slocum, R. McMaster, F. Kessler, and H. Howard. *Thematic Cartography and Geovisualization*. Pearson Prentice Hall, second edition edition, 2009.

[11] B. Speckmann and K. Verbeek. Necklace maps. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):881–889, 2010.

[12] P. Vaidya. Geometry helps in matching. *SIAM Journal on Computing*, 18(6):1201–1225, 1989.

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

[14] R. Veltkamp and M. Hagedoorn. State of the Art in Shape Matching. In *Principles of Visual Information Retrieval*, chapter 4, pages 87 – 115. Springer, 2001.

[15] A. Voorhees. A general theory of traffic movement. *Institute of Traffic Engineers*, pages 46–56, 1955.

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

[17] J. Wood, J. Dykes, and A. Slingsby. Visualisation of origins, destinations and flows with OD maps. *The Cartographic Journal*, 47(2):117–129, 2010.

[18] J. Wood, A. Slingsby, and J. Dykes. Visualizing the dynamics of London's bicycle hire scheme. *Cartographica*, 46:239–251, 2011.