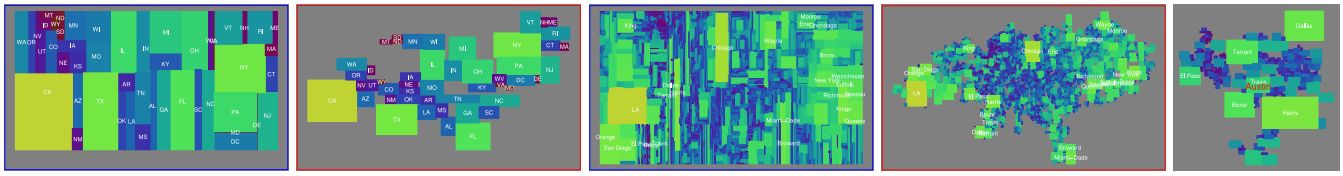
RecMap : Rectangular Map Approximations

Roland Heilmann* Bayer Technology
Daniel A. Keim† University of Konstanz
Christian Panse‡ University of Konstanz
Mike Sips§ University of Konstanz



(a) U.S. states (b) U.S. states (c) U.S. counties (d) U.S. counties (e) Texas

Figure 1: Results of *RecMap* — The area of each region corresponds to the number of people living there. The cartograms of Figures 1(a) and 1(c) contain no area error and no empty space error whereas in Figures 1(b), 1(d), and 1(e) no area error occurs and the shapes of the regions are preserved. The colormap indicates the number of people (yellow: high; brown: low).

ABSTRACT

In many application domains, data is collected and referenced by its geo–spatial location. Nowadays, different kinds of maps are used to emphasize the spatial distribution of one or more geo–spatial attributes. The nature of geo–spatial statistical data is the highly non–uniform distribution in the real world data sets. This has several impacts on the resulting map visualizations. *Classical area maps* tend to highlight patterns in large areas, which may, however, be of low importance. Cartographers and geographers used *cartograms* or *value–by–area maps* to address this problem long before computers were available. Although many automatic techniques have been developed, most of the value–by–area cartograms are generated manually via human interaction.

In this paper, we propose a novel visualization technique for geo–spatial data sets called *RecMap*. Our technique approximates a rectangular partition of the (rectangular) display area into a number of map regions preserving important geo–spatial constraints. It is a fully automatic technique with explicit user control over all exploration constraints within the exploration process. Experiments show that our technique produces visualizations of geo–spatial data sets, which enhance the discovery of global and local correlations, and demonstrate its performance in a variety of applications.

Keywords: Geographic Visualization, Information Visualization, Database and Data Mining Visualization

1 INTRODUCTION

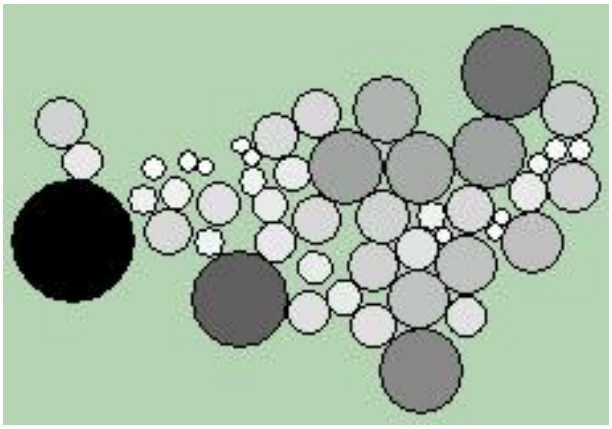
The information revolution is creating and publishing vast data sets, such as records of business transactions, environmental data, and census demographics. In many application domains, this data is collected and indexed by geo–spatial locations. The discovery of interesting patterns in geo–spatial databases is a key to turn this data into valuable information. Nowadays, different kinds of maps are used to emphasize the spatial distribution of one or more geo–spatial attributes. The nature of geo–spatial data is the highly non–uniform distribution in real world data sets which has several impacts on the resulting map visualizations. Imagine a data analyst who uses a traditional map, for example the U.S. map, and who wants to find very quickly potentially valuable information. The problem of using *classical area maps* is that they tend to highlight patterns in large areas, which may, however, be of low importance. It is mostly impossible for the data analyst to understand the presented geo–spatial information quickly, since this visual model does not address the human perception skills.

Cartographers and geographers used *cartograms* or *value–by–area maps* to address this problem long before computers were available. The basic idea of a cartogram is to distort a map by resizing its regions according to some external geography–related parameter. First hand–made cartograms can be found in [12, pp. 216–217]. Here, the area of a region corresponds to its population in 1958. (A detailed description of how to construct rectangular cartograms manually can be found in [12].) However, the manual construction of cartograms is a very difficult task because, on the one hand, we have to resize the regions according to their geo–spatial statistical values and, on the other hand, we have to take into account that the (original) shapes of the regions and their neighborhood relationships (*topology*) are preserved as much as possible.

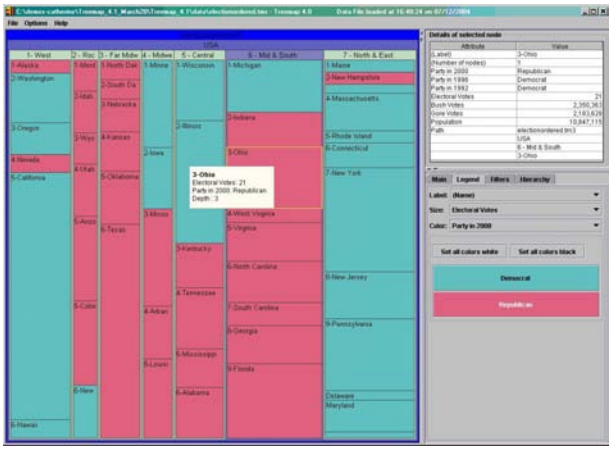
Consequently, the study of automated methods for drawing cartograms is of considerable interest. In the meantime, many automatic cartogram generation techniques have been developed (for an overview see [5, 7]). *Circular cartograms* [1] (see Figure 2(a)) ignore the shape of the input polygons completely and represent them as circles. In many cases, the area and topology constraints have to be relaxed, too. The general applicability of this technique is open to question. *TreeMaps* [6], which are a well–known information visualization technique, are an appropriate method to display data with a given hierarchic order. They divide the display area into rectangles such that the area of each rectangle corresponds to its statistical value. Figure 2(b) displays an example where *TreeMaps* are used for visualizing U.S. census data. To the best of our knowledge there do not exist any automatic procedures which compute the split hierarchy of the map. This work has to be done by the user

*roland.heilmann@bayertechnology.com
†keim@inf.uni-konstanz.de
‡panse@inf.uni-konstanz.de
§sips@inf.uni-konstanz.de

IEEE Symposium on Information Visualization 2004
October 10-12, Austin, Texas, USA
0-7803-8779-1/04/$20.00 ©2004 IEEE



(a) A screen capture of Dorling's *Circle Cartograms* [1] using the 2000 U.S. population data and the *cdv*-tool. [2]



(b) *Treemap* — interactively controlled method based on *TreeMaps* [10] – Visualizations of the presidential race in 2000.

Figure 2: Related work on *RecMap*

via interaction. Most of the techniques which have been presented so far do not take the shape and the topology of the map into account, e.g. [1], or the area error on continuous cartograms cannot be eliminated completely, e.g. [7].

2 Our Contribution

The idea of this work is to approximate familiar land covering map region shapes by rectangles and to find a partition of the available screen space where the areas of these rectangular regions are proportional to given statistical values. In order to support the understanding of the information represented by a cartogram we try to place the rectangles as close as possible to their original positions and as close as possible to their neighbors. We define two variants of this optimization problem and present two corresponding (heuristic) algorithms which generate space filling partitions of the screen space with respect to the given geo–locations. Both algorithms construct cartograms where the area of each rectangle of the

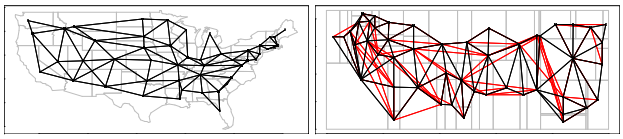


Figure 3: The Figures show the adjacency graphs of the U.S. map (left) and a corresponding map partition (right). The red colored segments indicate the topology error.

cartogram is proportional to its area within the map. The difference between these heuristics is that the first method does not allow empty space, whereas the second one preserves the shapes of the polygons.

The remainder of this paper is organized as follows: Section 3 is devoted to a formal description of the (variants of the) *map partition problem* or *cartogram problem*. In Section 4, we present two heuristic solution procedures. The efficiency of our new approach as well as some application examples are shown in Section 5.

3 Problem Definition

In this section, we give a formulation of the problem of determining a near–optimal cartogram $\mathscr{P} = \{p_1, \ldots, p_R\}$ for a given map $\overline{\mathscr{P}} = \{\overline{p}_1, \ldots, \overline{p}_R\}$ consisting of R *polygons* or *regions* and vector $Z = (z_r)_{r=1,\ldots,R}$ of *spatial data values* $z_r \geq 0$ with $\sum_{r=1}^{R} z_r = 1$. For this, we first refer to the constraints which have to be met during the optimization process. Hereafter, we turn to the single components of the objective function.

3.1 Constraints

When determining $\mathscr{P}$ we can choose among several possibilities of representing the regions of $\overline{\mathscr{P}}$. We have decided to use rectangular polygons as in this way the expressiveness of $\mathscr{P}$ is not impaired by insignificant details of the shapes of the polygons of $\overline{\mathscr{P}}$. As indicated before, we name this type of cartogram *rectangular map*. Hence, we have to meet the following constraints in any case:

- $\mathscr{P}$ is planar,
- each polygon $p \in \mathscr{P}$ is a rectangle, and
- each polygon $p \in \mathscr{P}$ is neighbor of at least one different polygon $p' \in \mathscr{P}$.

A cartogram $\mathscr{P}$ obeying these constraints is called *feasible*. The set of feasible cartograms is denoted by $\mathscr{M}$.

3.2 Objective function

The quality of $\mathscr{P}$ depends on two aspects: First, we have to evaluate whether the polygons of $\overline{\mathscr{P}}$ can be easily recognized in $\mathscr{P}$. Second, the areas of the polygons of $\mathscr{P}$ have to reflect the geo–spatial data values given by Z. In general, these requirements represent conflicting goals. Based on these aspects, we use five criteria in order to evaluate the quality of $\mathscr{P}$. These criteria, which correspond to the components of the objective function, are presented in the following.

Area The quality of $\mathscr{P}$ w.r.t. the criterion "area" is measured by the *area error* $A(\mathscr{P}) = A(\mathscr{P}, Z)$ with

$$A(\mathscr{P}) := \frac{1}{R} \cdot \sum_{r=1}^{R} \frac{|\varepsilon_r - z_r|}{z_r}. \quad (1)$$

$A(\mathscr{P})$ equals the average relative deviation of the actual share $\varepsilon_r := a(p_r)/A_f(\mathscr{P})$ of the area $a(p_r)$ of a polygon p_r in the *filled space* or *filled area* $A_f(\mathscr{P}) := \sum_{r=1}^{R} a(p_r)$ of $\mathscr{P}$ from its desired share z_r in $A_f(\mathscr{P})$.

Shape The *shape error* $S(\mathscr{P}) = S(\mathscr{P}, \overline{\mathscr{P}})$ reflects the average relative deviation of the shape of a polygon $p_r \in \mathscr{P}$ from that of its corresponding polygon $\overline{p}_r \in \overline{\mathscr{P}}$ and is determined as follows:

$$S(\mathscr{P}) := \frac{1}{R} \cdot \sum_{r=1}^{R} \frac{|s(p_r) - s(\overline{p}_r)|}{s(\overline{p}_r)}. \quad (2)$$

The shape $s(\overline{p}_r)$ or $s(p_r)$ of a polygon $\overline{p}_r \in \overline{\mathscr{P}}$ or $p_r \in \mathscr{P}$ is measured by the ratio of its maximum extension in horizontal direction and its maximum extension in vertical direction, respectively.

Topology The *topology error* $T(\mathscr{P})$ is an indicator of the deviation of the neighborhood relationships given by $\mathscr{P}$ from those given by $\overline{\mathscr{P}}$. To obtain $T(\mathscr{P})$, we first have to compute the *adjacency graphs* or *pseudo dual graphs* $\overline{G}_a$ and G_a of $\overline{\mathscr{P}}$ and $\mathscr{P}$, respectively. An adjacency graph reflects the neighborhood relationships between the polygons of a polygon mesh (cf. [9], p. 267). To obtain that graph, we first introduce a vertex for each polygon of the polygon mesh. Next, for each pair of neighbored polygons, we add an edge between the corresponding vertices.

$T(\mathscr{P}) = T(\mathscr{P}, \overline{\mathscr{P}})$ with

$$T(\mathscr{P}) := \frac{|\overline{E}_a \backslash E_a| + |E_a \backslash \overline{E}_a|}{|\overline{E}_a \cup E_a|}, \quad (3)$$

where $\overline{E}_a$ and E_a denote the set of edges of $\overline{G}_a$ and G_a, respectively, reflects the number of neighborhood relationships being solely contained in one of both polygon meshes and is normalized to the interval $[0,1]$. An example for the calculation of $T(\mathscr{P})$ is given by Figure 3.

Relative polygon positions An important criterion for the recognizability of the polygons in $\mathscr{P}$ are their relative positions. But since they are only partially reflected by the adjacency graph we use the (*relative*) *position error* $R(\mathscr{P}) = R(\mathscr{P}, \overline{\mathscr{P}})$ with

$$R(\mathscr{P}) := \frac{2}{R \cdot (R-1)} \cdot \frac{1}{180^\circ} \cdot \sum_{r=1}^{R-1} \sum_{\rho=r+1}^{R} \alpha_{r,\rho}, \quad (4)$$

which is normalized to the interval $[0,1]$. $\alpha_{r,\rho} := \arccos(\vec{\overline{u}}_{r,\rho} \cdot \vec{u}_{r,\rho})/(|\vec{\overline{u}}_{r,\rho}| \cdot |\vec{u}_{r,\rho}|)$ measures the deviation of the relative positions of polygons p_r and p_ρ from those of $\overline{p}_r$ and $\overline{p}_\rho$ with the help of vectors $\vec{\overline{u}}_{r,\rho} = c(\overline{p}_\rho) - c(\overline{p}_r)$ and $\vec{u}_{r,\rho} = c(p_\rho) - c(p_r)$ where $c(\overline{p}_r)$ and $c(p_r)$ stand for the centers of gravity of $\overline{p}_r$ and p_r, respectively.

Empty space As we make use of rectangular maps it might happen that $\mathscr{P}$ contains "holes" or *empty space* which comprises those areas which are completely surrounded by filled space, i.e. by polygons of $\mathscr{P}$. Consequently, we also measure the quality of $\mathscr{P}$ by the *empty space error* $E(\mathscr{P})$ with

$$E(\mathscr{P}) := \frac{A_t(\mathscr{P}) - A_f(\mathscr{P})}{A_t(\mathscr{P})}, \quad (5)$$

which equals the share of empty area in the total area $A_t(\mathscr{P})$ of $\mathscr{P}$. $A_t(\mathscr{P})$ stands for the space being enclosed by the boundary of $\mathscr{P}$. Again, $E(\mathscr{P})$ is normalized to the interval $[0,1]$.

3.3 Formulation of the optimization problem

To give the user full control over the visualization goals we have developed two variants of the map partition problem each of them focussing on different components of objective function $F(\mathscr{P}) = (A(\mathscr{P}), S(\mathscr{P}), T(\mathscr{P}), R(\mathscr{P}), E(\mathscr{P}))$.

Variant 1 (*MP*1) Since one of the most important aspects w.r.t. the expressiveness of cartograms is that spatial data is represented by area, we require that $A(\mathscr{P})$ equals zero. In order to use the full screen space, we demand that $E(\mathscr{P})$ equals zero, too. Hence, using the constraints and the components of F which have been introduced above, we can state the first variant of the map partition problem being considered in this paper as the following *vector minimum problem*:

$$\text{Min.} \quad F(\mathscr{P}) \quad (6)$$

$$\text{s.t.} \quad \mathscr{P} \in \mathscr{M},\ A(\mathscr{P}) = 0,\ \text{and}\ E(\mathscr{P}) = 0. \quad (7)$$

Variant 2 (*MP*2) Like for (*MP*1), we demand that no area error occurs. Second, in order to take the recognizability of the polygons into account, we do not allow any shape error. Consequently, we obtain the following optimization problem:

$$\text{Min.} \quad F(\mathscr{P}) \quad (8)$$

$$\text{s.t.} \quad \mathscr{P} \in \mathscr{M},\ A(\mathscr{P}) = 0,\ \text{and}\ S(\mathscr{P}) = 0. \quad (9)$$

It is likely that (*MP*1) and (*MP*2) represent $\mathscr{NP}$–hard optimization problems.

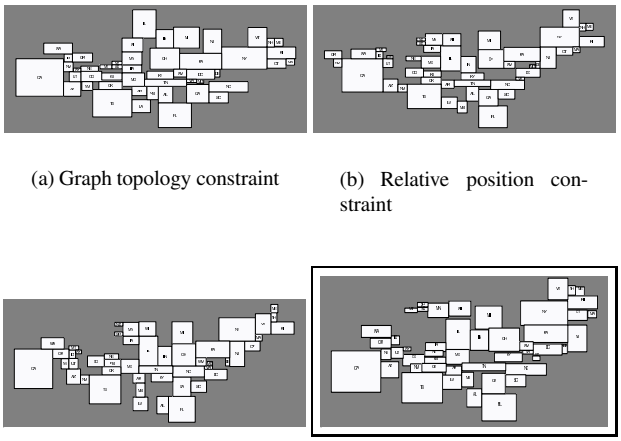
4 The *RecMap* Algorithm

In the following, we are going to present heuristic solution procedures for both variants of the map partition problem. First, we refer to a heuristic for (*MP*1). Hereafter, we present a method which computes a near–optimal solution for (*MP*2).

To obtain cartograms of high quality, we repeat the construction of cartograms using a *genetic algorithm* (cf. [4], 2000) which guides the optimization process. In each iteration of this *meta heuristic*, a set or *generation* of cartograms or *individuals* is generated. An individual is characterized by three aspects: the *genotype*, the *construction algorithm*, and the *phenotype*. The genotype stands for the information needed to generate the corresponding phenotype using a certain construction algorithm. In our context, the genotype corresponds to an array of nonnegative integers and the phenotype to a (feasible) cartogram.

The individuals of a generation are evaluated by means of a weighted objective function $\widehat{F}$ which is derived from F. Then, we *select* a predefined number of best individuals and determine the next generation out of their genotypes by applying *replication* and *mutation*. This process is repeated until a predefined number of generations has been generated or a given amount of time has elapsed. The best cartogram which has been found so far is returned.

The weights of $\widehat{F}$ can be set by the user according to her or his visualization goals. In this way, the user gains control over the visualization process and result. The effect of different weights on the



(a) Graph topology constraint

(b) Relative position constraint

(c) Empty space constraint

(d) A combination of all constraints

Figure 4: Cartograms $\mathscr{P}$ resulting from different weights for the components of F

resulting cartograms is demonstrated in Figure 4 w.r.t. our heuristic for ($MP2$). For example, Figure 4(a) shows the cartogram which is obtained if the weight for $T(\mathscr{P})$ is set to one and the other weights are set to zero. (Figures 4(b) and 4(c) have to be interpreted in an analogous manner.) The cartogram of Figure 4(d) is obtained if all weights are set to one.

4.1 Variant 1

4.1.1 Basic idea

In order to achieve $E(\mathscr{P}) = 0$, our heuristic is based on the procedure of [8] which — starting with a single rectangle — computes a sequence of *partial cartograms* $\widetilde{\mathscr{P}}$ by adding a rectangle in each step in such a way that no empty space can occur. In the following, we first give a detailed description of this basic procedure which serves as the construction algorithm within our genetic algorithm.

4.1.2 Initialization step

In the *initialization step*, we first draw the initial partial cartogram which consists of a rectangle, denoted by p_1, with an horizontal extension of $\max_{r=1,\ldots,R} \max_{i=1,\ldots,\overline{n}_r} \overline{x}_i^r - \min_{r=1,\ldots,R} \min_{i=1,\ldots,\overline{n}_r} \overline{x}_i^r$ and a vertical extension of $\max_{r=1,\ldots,R} \max_{i=1,\ldots,\overline{n}_r} \overline{y}_i^r - \min_{r=1,\ldots,R} \min_{i=1,\ldots,\overline{n}_r} \overline{y}_i^r$. ($\overline{n}_r$ stands for the number of nodes of $\overline{p}_r$ and $(\overline{x}_i^r, \overline{y}_i^r)$ for the position of the ith vertex of $\overline{p}_r$ in clockwise order.) Second, we compute the center of gravity $c(\overline{p}_r)$ for each polygon $\overline{p} \in \overline{\mathscr{P}}$ and plot this point into the starting rectangle. In the following, the polygons of $\overline{\mathscr{P}}$ are represented by their centers of gravity.

4.1.3 Main step

In the *main step*, we perform a sequence of so–called *splits*. Each split refers to a rectangular polygon $p \in \widetilde{\mathscr{P}}$, which contains at least two centers of gravity, and divides it into two new rectangular polygons, each of them containing at least one center of gravity. In this way, we construct a sequence of partial cartograms $\widetilde{\mathscr{P}}$ with no empty space error. Each $p \in \widetilde{\mathscr{P}}$ represents the aggregation of those polygons of $\overline{\mathscr{P}}$ which correspond to the centers of gravity $c(\overline{p}_r)$ being contained in p (i.e. $c(\overline{p}_r) \in p$). The main step ends, when each rectangle contains exactly one

center of gravity and hence no further split can be done. Consequently, after $R-1$ splits we obtain a partial cartogram $\widetilde{\mathscr{P}}$ with R polygons. This final (partial) cartogram corresponds to a (complete) cartogram: $\mathscr{P} := \widetilde{\mathscr{P}}$. The polygons of $\mathscr{P}$ have to be re–numbered because, as indicated before, a rectangle $p \in \mathscr{P}$ represents that polygon $\overline{p}_r \in \overline{\mathscr{P}}$ which corresponds to the single center of gravity being included in p. Therefore, p gets the index r.

We differ between two types of splits: a *horizontal split* and a *vertical split*. A horizontal (vertical) split introduces a horizontal (vertical) *splitting line* into the rectangle $p \in \widetilde{\mathscr{P}}$ to be split, which results in two new polygons p' and p'' with p' being below (left) of p''. Splitting is done in such a way that, after each split, we have for the current partial cartogram $\widetilde{\mathscr{P}}$

$$\frac{a(p_r)}{\sum_{\rho=1}^{|\widetilde{\mathscr{P}}|} a(p_\rho)} = \sum_{c(\overline{p}_r)\in p} z_r \qquad (r = 1,\ldots,|\widetilde{\mathscr{P}}|), \tag{10}$$

i.e. the area of $p_r \in \widetilde{\mathscr{P}}$ is proportional to the sum $\sum_{c(\overline{p}_r)\in p} z_r$ of the spatial data values of the polygons $\overline{p}_r \in \overline{\mathscr{P}}$ being associated with $c(\overline{p}_r) \in p$ ($r = 1,\ldots,|\widetilde{\mathscr{P}}|$).

We try to split a polygon $p \in \widetilde{\mathscr{P}}$ as equally as possible: If we do a horizontal (vertical) split, we scan the points $c(\overline{p}_r) \in p$ from bottom to top (left to right), and add them to p' until we have

$$\sum_{c(\overline{p}_r)\in p'} z_r \geq \frac{1}{2} \cdot \sum_{c(\overline{p}_r)\in p} z_r. \tag{11}$$

Those centers of gravity of p which have not been added to p' are added to p''. If p contains two centers of gravity, we stop after having scanned the first. Provided that we perform a horizontal or vertical split of $p_r \in \widetilde{\mathscr{P}}$, the splitting line is placed into p_r such that the height or the breadth of p' equals

$$\frac{\sum_{c(\overline{p}_r)\in p'} z_r}{\sum_{c(\overline{p}_r)\in p} z_r} \cdot (y_3^r - y_1^r) \quad \text{or} \quad \frac{\sum_{c(\overline{p}_r)\in p'} z_r}{\sum_{c(\overline{p}_r)\in p} z_r} \cdot (x_3^r - x_1^r), \tag{12}$$

respectively.

4.1.4 The *RecMap* –algorithm for Variant 1

A major drawback of the procedure described in the previous section is its rigidity. This means, that the polygons resulting from a horizontal split have to be split vertically in any case and vice versa. But in this way, no special attention is paid to the shapes of the polygons and the neighborhood relationships between them. For example, if the majority of the polygons $\overline{p} \in \overline{\mathscr{P}}$ possesses a longish shape (i.e. $s(\overline{p}) < 1$) the procedure might lead to seriously deformed cartograms, i.e. cartograms $\mathscr{P}$ with high values for $S(\mathscr{P})$ and $T(\mathscr{P})$. In such a case, it would have been indicated to prefer vertical splits.

This drawback can be avoided by using *split sequences*. For example, let a cartogram $\mathscr{P}$ be obtained by performing a horizontal split and two vertical splits afterwards. If we associate a horizontal split

with 0 and a vertical split with 1, we get the split sequence $(0,1,1)$. This split sequence can be conceived as the genotype of $\mathscr{P}$. In general, the genotype of a cartogram is a vector $(I_\lambda)_{\lambda=1,\ldots,R-1}$ of (binary) values $I_\lambda \in \{0,1\}$.

To use the construction algorithm described above w.r.t. a given split sequence $(I_\lambda)_{\lambda=1,\ldots,R-1}$, it has to be adapted accordingly. For this, we introduce variable λ which stands for the split position which is currently considered. I_λ represents the splitting type to be chosen for the λth split. At the end of the algorithm, λ equals $R-1$. The adapted construction algorithm is given by Algorithm 1.

STEP 1 (*initialization step*)
$\widetilde{P} := \{p_1\}$; $\mathscr{S} := \{p_1\}$; $\lambda := 1$
STEP 2 (*main step*)
WHILE $\mathscr{S} \neq \emptyset$ **DO**
  Remove p from $\mathscr{S}$
  IF $|\{\overline{p}_r \in \overline{\mathscr{P}} | c(\overline{p}_r) \in p\}| > 1$ **THEN**
    IF $I_\lambda = 0$ **THEN**
      Split p horizontally into p' and p''
    ELSE
      Split p vertically into p' and p''
    $\mathscr{S} := \mathscr{S} \cup \{p''\}$; $\mathscr{S} := \mathscr{S} \cup \{p'\}$
    $\widetilde{\mathscr{P}} := \widetilde{\mathscr{P}} \backslash \{p\}$; $\widetilde{\mathscr{P}} := \widetilde{\mathscr{P}} \cup \{p', p''\}$
    $\lambda := \lambda + 1$
RETURN $\widetilde{\mathscr{P}}$

Algorithm 1: Construction procedure for the $MP1$ heuristic

4.2 Variant 2

4.2.1 Basic idea

In literature, we find an optimization problem in the context of *inner–plant layout planning* (cf. [9], pp. 255) which shows certain similarities to ($MP2$). This *layout problem* can be roughly stated as follows: We are given a rectangular site, a set of machines (being described by their ground plans), and the amounts of material which have to be transported between them. The objective is to find a *layout*, i.e. a plan with the positions of the machines on the site, such that the transportation costs are minimized, i.e. that machines with a high transportation intensity in between are located as close as possible. The map partition problem considered here is similar to the layout problem in so far as we are also given a set of rectangular areas (i.e. polygons) which have to be placed such that the distances between them are taken into consideration.

The construction algorithm of our heuristic for ($MP2$) relies on two ideas of the procedure of [3] which is derived from a graph theoretical model of the layout problem. First, in the *initialization step*, we choose a specific polygon, called the *core polygon* p^c, to be the center of the *layout* or *cartogram*. Second, in the *main step*, we construct a sequence of *partial layouts* or *partial cartograms* $\widetilde{\mathscr{P}}$, i.e. starting with p^c, the remaining $R-1$ polygons are placed around it one after the other until we have found a (complete) cartogram $\mathscr{P} := \widetilde{\mathscr{P}}$.

4.2.2 Initialization step

As the area error and the shape error ought to be zero, the breadth $b(p_r)$ and height $h(p_r)$ of each polygon $p_r \in \mathscr{P}$ are given by

$$b(p_r) := \sqrt{z_r \cdot \sum_{r=1}^{R} a(\overline{p}_r) \cdot s(\overline{p}_r)} \text{ and} \quad (13)$$

$$h(p_r) := \sqrt{\frac{z_r \cdot \sum_{r=1}^{R} a(\overline{p}_r)}{s(\overline{p}_r)}}, \quad (14)$$

respectively. The core polygon p^c is determined with the help of an extension $\overline{G}_{a_x}$ of $\overline{G}_a$ which is obtained by introducing an additional node $R+1$ for the outer region of $\overline{\mathscr{P}}$. p^c corresponds to a polygon $\overline{p}_r$ which has the maximum distance $d_{r,R+1}$ from the outer region $\overline{p}_{R+1}$ where the distance between two polygons is measured by the minimum number of edges between their corresponding nodes in $\overline{G}_{a_x}$.

4.2.3 Main step

As indicated before, the main step consists of $R-1$ partial steps. In each of these steps, we choose an index r among the set of indices of those polygons which have not yet been created and added to $\widetilde{\mathscr{P}}$. Let Q denote the set of indices of the polygons $p_r \in \widetilde{\mathscr{P}}$. Then, the index of the newly created polygon has to be taken from $\{1,\ldots,R\}\backslash Q$. Since the maintenance of the neighborhood relationships of $\overline{\mathscr{P}}$ is of high importance w.r.t. the recognizability of the polygons of $\mathscr{P}$, we have a look at $\overline{G}_a$ in order to determine the index r' of the polygon to be added next. We demand that the corresponding polygon $\overline{p}_{r'} \in \overline{\mathscr{P}}$ is a neighbor of at least one of those polygons $\overline{p}_r \in \overline{\mathscr{P}}$ the indices of which are contained in Q because otherwise, the adjacency graph of the resulting partial cartogram $\widetilde{\mathscr{P}}$ would not be connected and we could no longer guarantee $\mathscr{P} \in \mathscr{M}$ for the final cartogram $\mathscr{P}$. Let $\mathscr{N}(\overline{p}_r)$ denote the set of neighbors of $\overline{p}_r$ in $\overline{\mathscr{P}}$, i.e. $\mathscr{N}(\overline{p}_r) = \{\overline{p}_\rho | (r,\rho) \in \overline{E}_a\}$. If there are several indices which could be selected, we choose the lowest one. Thus, we set

$$r' := \min\{r \in \{1,\ldots,R\}\backslash Q | \overline{p}_r \in \bigcup\nolimits_{\rho \in Q} \mathscr{N}(\overline{p}_\rho)\}. \quad (15)$$

After the determination of r' we have to decide where to place the corresponding polygon $p_{r'}$. In general, there exists an infinite number of possible positions for $p_{r'}$. In order to keep the computational time low, we have to restrict ourselves to a finite subset.

Pretests have revealed that the following procedure is favorable: We scan the edges e of the boundary $\mathscr{E}$ of $\widetilde{\mathscr{P}}$ and determine a set Π_e of possible positions for $p_{r'}$ w.r.t. e. For example, we add the end points and the middle point of e to Π_e as possible positions of the lower left corner of $p_{r'}$. To keep the number of possible positions low and to exclude infeasible positions, each position $(x,y) \in \Pi_e$ is checked within a multi–stage test. For instance, we remove those positions from Π_e which cause a violation of the planarity or which could lead to unacceptable high values of $T(\mathscr{P})$ or $R(\mathscr{P})$.

Subsequent to the determination of the sets Π_e we select the best position $(x^\star, y^\star)$ and create $p_{r'}$ at that position. $(x^\star, y^\star)$ is found as follows: Let Π denote the set of all feasible positions which have been found so far. For each position $(x,y) \in \Pi$, we temporarily extend $\widetilde{\mathscr{P}}$ by adding the newly created polygon $p_{r'}$ at (x,y) and compute a weighted sum $\widehat{F}(\widetilde{\mathscr{P}})$ of the values of the components of F. The position associated with the minimum value of $\widehat{F}$ equals $(x^\star, y^\star)$.

4.2.4 The *RecMap* –algorithm for Variant 2

As the order in which the polygons are added to $\widetilde{\mathscr{P}}$ is of high importance we have to encode this information by the genotype of $\mathscr{P}$ which equals a vector $(I_\lambda)_{\lambda=1,\ldots,R}$ of values $I_\lambda \in \{1,\ldots,R\}$ with $I_\lambda \neq I_{\lambda'}$ for $\lambda \neq \lambda'$ $(\lambda,\lambda' \in \{1,\ldots,R\})$. For the use within our genetic algorithm–based heuristic, we have to adapt the choice of r' accordingly. This means, that among the set of indices of those polygons which are neighbors of polygons $\overline{p}_r$ with $r \in Q$, we select that index r' which is the first of them in I. Hence, (15) has to be modified as follows:

$$r' := \min\{\lambda \in \{1,\ldots,R\} | I_\lambda \in \{1,\ldots,R\}\backslash Q \text{ and } \overline{p}_{I_\lambda} \in \bigcup_{\rho \in Q} \mathscr{N}(\overline{p}_\rho)\}. \quad (16)$$

The construction algorithm is given by Algorithm 2.

STEP 1 (*initialization step*)
Create $\overline{G}_{a_x}$
$r' := \min\{r \in \{1,\ldots,R\} | d_{r,R+1} = \max_{\rho=1,\ldots,R} d_{\rho,R+1}\}$
Create $p_{r'}$ at $(0,0)$; $\widetilde{\mathscr{P}} := \{p_{r'}\}$; $Q := \{r'\}$; $p^c := p_{r'}$
STEP 2 (*main step*)
WHILE $Q \neq \{1,\ldots,R\}$ **DO**
  $r' := \min\{r \in \{1,\ldots,R\}\backslash Q | \overline{p}_r \in \bigcup_{\rho \in Q} \mathscr{N}(\overline{p}_\rho)\}$
  $Q := Q \cup \{r'\}$
  $\Pi := \emptyset$
  FOR $e \in \mathscr{E}$ **DO**
    Determine Π_e; $\Pi := \Pi \cup \Pi_e$
  $\widehat{F}^\star := \infty$
  FOR $(x,y) \in \Pi$ **DO**
    Create $p_{r'}$ at (x,y); $\widetilde{\mathscr{P}} := \widetilde{\mathscr{P}} \cup \{p_{r'}\}$
    IF $\widehat{F}(\widetilde{\mathscr{P}}) < \widehat{F}^\star$ **THEN**
      $(x^\star,y^\star) := (x,y)$; $\widehat{F}^\star := \widehat{F}(\widetilde{\mathscr{P}})$
    $\widetilde{\mathscr{P}} := \widetilde{\mathscr{P}}\backslash\{p_{r'}\}$
  Create $p_{r'}$ at $(x^\star,y^\star)$; $\widetilde{\mathscr{P}} := \widetilde{\mathscr{P}} \cup \{p_{r'}\}$
RETURN $\widetilde{\mathscr{P}}$

Algorithm 2: Construction procedure for the $MP2$ heuristic

5 Efficiency & Application

The algorithms described above were implemented in *ANSI–C* using the `O4` compiler option and run on the operating systems Microsoft Windows and UNIX. The tests were performed on a 1.5 GHz Intel Xeon server with 4 GB main memory under Linux (only 1 MB was needed by the algorithms). In this section, we will show some applications using the U.S. census data base and U.S. election data.

Efficiency Figure 5 shows, for each generation, the best cartogram which has been found so far by the heuristic for ($MP2$) and Figure 6 illustrates the respective values of the errors. The values for the ($MP1$) approach are illustrated, too. From these scatterplots we can conclude that the heuristic for ($MP2$) is more time consuming than that for ($MP1$). Since *RecMap* gives us a useful visualization even after the first iteration the interactive exploration of the data is guaranteed (see the cartogram yielded after Step 1 in Figure 5).

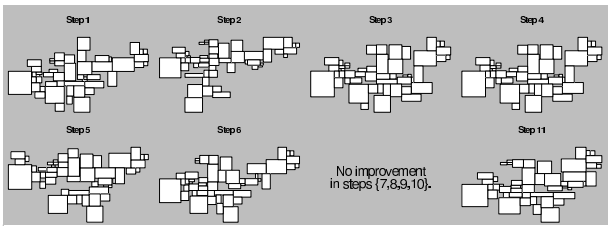


Figure 5: This Figure demonstrates the continuous improvement of the feasible solutions for ($MP2$) with increasing number of generations. The computation time for each iteration equals 5 seconds.

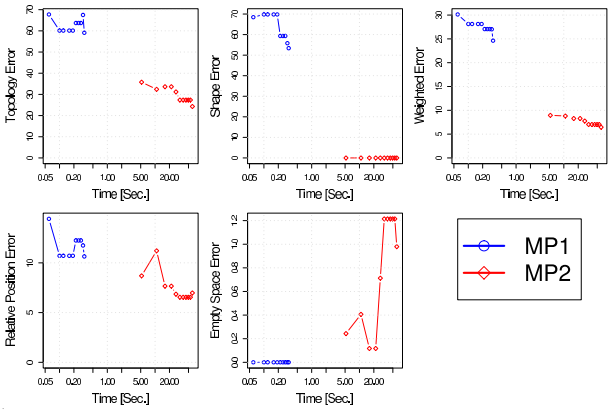


Figure 6: The scatterplots display the errors over time yielded by the ($MP1$) and ($MP2$) heuristics for U.S. state level. Note that the time axes are logarithmically scaled. The whole computation time for 10 iterations equals 0.33 seconds for ($MP1$) and 55 seconds for ($MP2$).

U.S. election analysis We ran the algorithms on a number of example data sets. Figures 7(b) and 7(c) show the U.S. electoral voters corresponding to the presidential election result in 2000.

U.S. census analysis We have applied *RecMap* to the population data of the U.S. census data set. In Figure 8(a) we show the original map on state and county level. On all maps of Figure 8 a unipolar colormap (see Figure 8(h)) is used to indicate the number of people. In Figures 8(b) and 8(c), highly populated regions, e.g. L.A. and Chicago, are made clearly visible. Figures 8(d)–8(g) show rectangular maps for California, New York state, and Texas using *RecMap* for Variants 1 and 2. Please note that for all rectangular cartograms the area error is zero.

RecMap has been integrated into our *CartoView–System* together with other information visualization techniques for geo–spatial data which have been introduced in the past [7, 8]. The interaction allows us to make the exploration process more efficient and effective.

6 Conclusion & Future Work

In this study we have analyzed and discussed the problem of efficient map partitioning and have proposed two automatic, scalable,

and flexible algorithms called *RecMap* for generating rectangular map partitions. Here, the user has an explicit control of all visualization constraints. Our approach is novel because its features (no area error, explicit control of shape, topology, empty space, and relative position constraints) are not provided by previous approaches. This new technique enables interactive views of detail at various levels to find very fast interesting patterns or subsets.

The experiments show that our algorithms offer good results for a variety of applications, and their speed even allows an interactive display. Further research could comprise the combination of our approach and other visualization techniques such as pixel–oriented techniques (where the pixels could be placed directly into their corresponding map partitions). Such a combination would allow to visualize areas with high information density. Additional material (e.g. an executable file) and ongoing work can be found on our web site `http://dbvis.inf.uni-konstanz.de/projects/RecMap`.

Acknowledgment We thank Stephen C. North from the AT&T Shannon Laboratory in Florham Park, New Jersey, U.S., for encouraging this research. We thank Catherin Plaisant for preparing the *Treemap* picture in Figure 2(b). This work was partially funded by the Information Society Technologies program of the European Commission, Future and Emerging Technologies under the IST–2001–33058 PANDA project (2001–2004).

REFERENCES

[1] Daniel Dorling. *Area Cartograms: Their Use and Creation*. Department of Geography, University of Bristol, England, 1st edition, 1996.

[2] J.A. Dykes. Cartographic visualization: Exploratory spatial data analysis with local indicators of spatial association using tcl/tk and cdv. *The Statistician*, 47(3):485–497, 1998.

[3] L.R. Foulds. *Graph Theory Applications*. Springer, Berlin, 1992.

[4] M. Gen and R. Cheng. *Genetic Algorithm and Engineering Optimization*. Wiley, New York, 2000.

[5] Donald H. House and Christopher J. Kocmoud. Continuous cartogram construction. In *IEEE Visualization, Research Triangle Park, NC*, pages 197–204, 1998.

[6] B. Johnson and B. Shneiderman. Treemaps: A space-filling approach to the visualization of hierarchical information. In *Proc. Visualization '91 Conf*, pages 284–291, 1991.

[7] Daniel A. Keim, Stephen C. North, and Christian Panse. CartoDraw: A fast algorithm for generating contiguous cartograms. *Transactions on Visualization and Computer Graphics*, 10(1):95–110, January/February 2004.

[8] Daniel A. Keim, Stephen C. North, Christian Panse, and Jörn Schneidewind. Efficient cartogram generation: A comparison. In *InfoVis 2002, IEEE Symposium on Information Visualization, Boston, Massachusetts*, pages 33–36, October 2002.

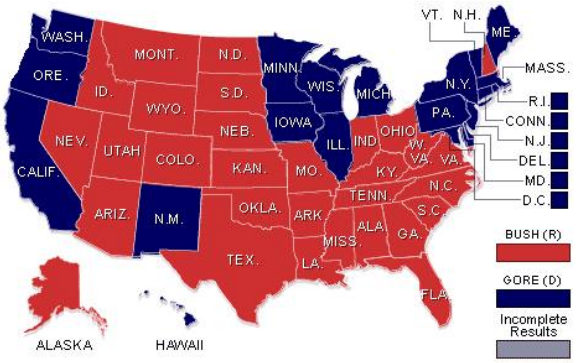
[9] K. Neumann. *Produktions– und Operations–Management*. Springer, Berlin, 1996.

[10] Kent Norman, Ben Shneiderman, Catherine Plaisant, Evan Golub, Chris North, Gunjan Dang, Egemen Tanin, , and Haixia Zhao. User interfaces for the u.s. bureau of census online survey interfaces and data visualization, Mar, 30th 2004. `http://www.cs.umd.edu/projects/hcil/census/`.

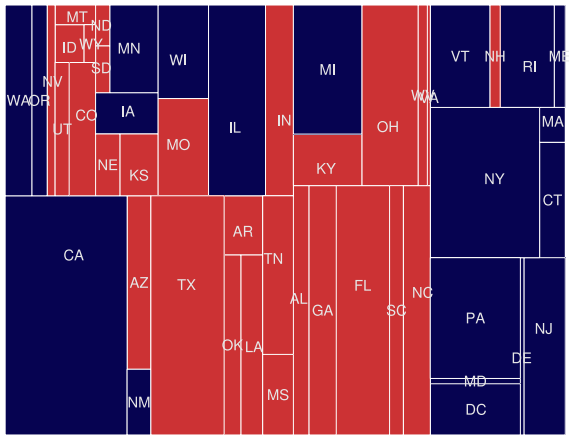
[11] The New York Times on the Web. The 2000 election, Mar, 30th 2004. `http://www.nytimes.com/specials/election2000/results-pres.html`.

[12] Erwin Raisz. *Principles of Cartography*. McGraw-Hill, New York, 1962.

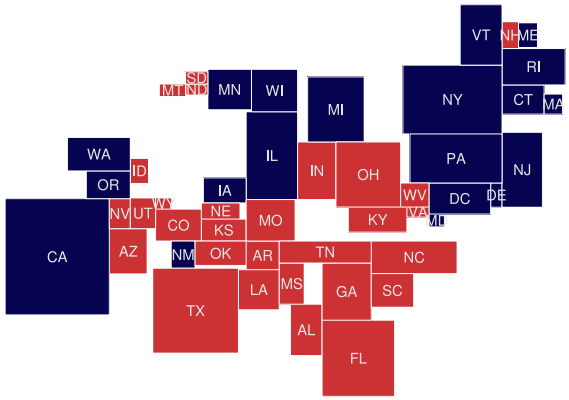
[13] Marc van Kreveld and Bettina Speckmann. On rectangular cartograms. In *20th European Workshop on Computational Geometry, Seville*, March 2004. `http://www.us.es/ewcg04/Articulos/Speckmann.ps`.



(a) Traditional map which can be accessed at [11]

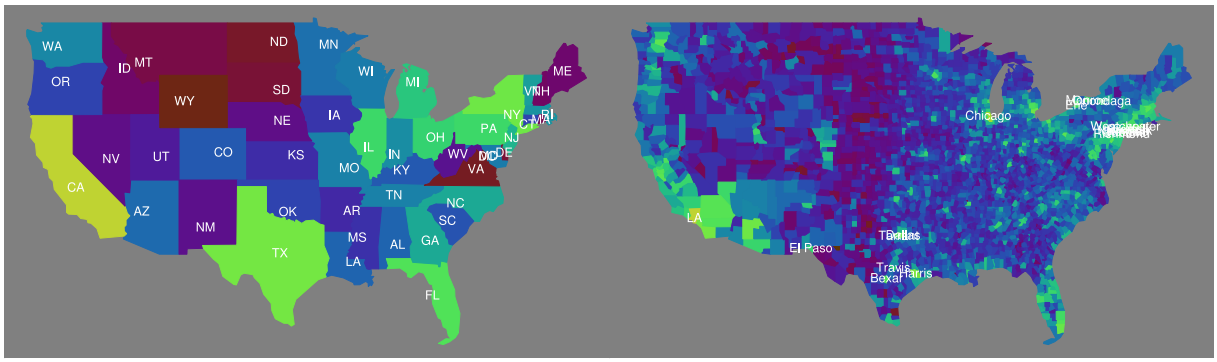


(b) *RecMap* for (*MP*1)

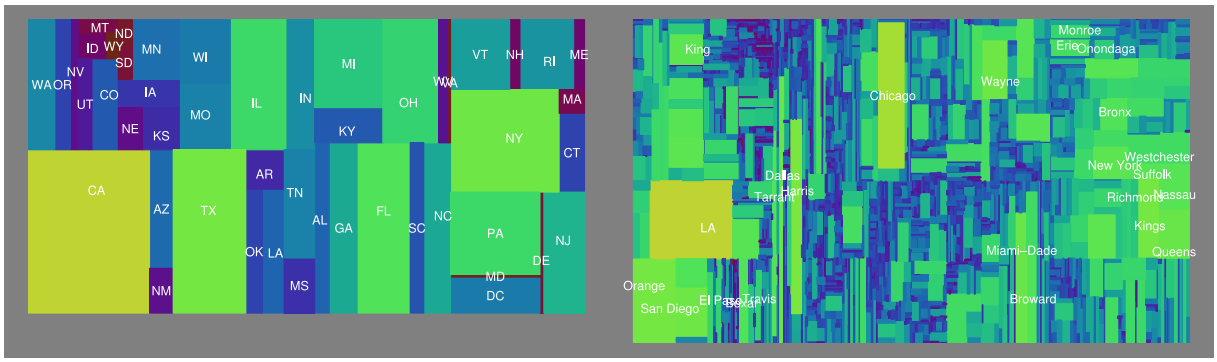


(c) *RecMap* for (*MP*2)

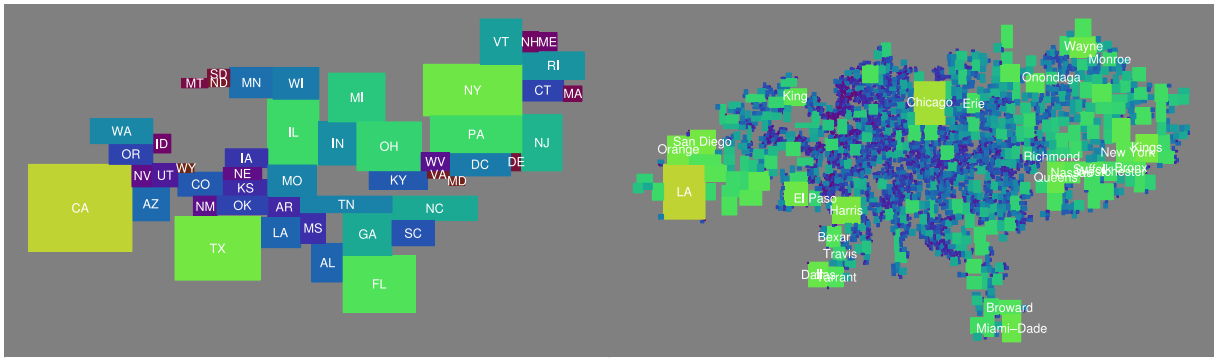
Figure 7: Visualizations of the presidential race in 2000. The areas of Figures 7(b) and 7(c) correspond to the numbers of electoral voters. The red and the blue color depict which candidate has won each state. The candidate who covers most of the area of all polygons in Figures 7(b) and 7(c) has won the election.



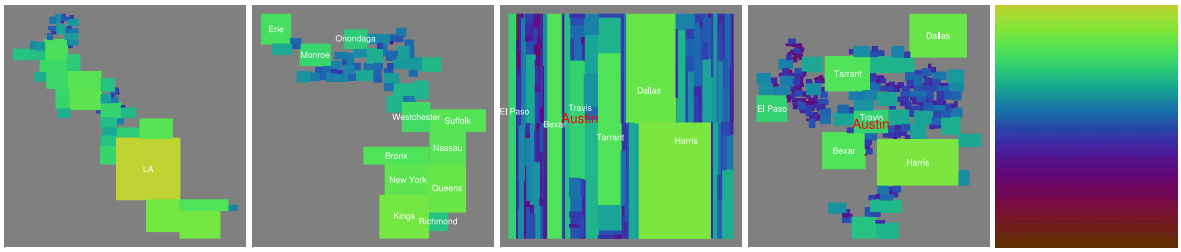
(a) Traditional U.S. map on state and county level



(b) Results of *RecMap* (Variant 1) using U.S. census population data on U.S. state and county level



(c) Results of *RecMap* (Variant 2) using U.S. census population data on U.S. state and county level



(d) CA using Variant 2 (e) NY using Variant 2 (f) TX using Variant 1 (g) TX using Variant 2 (h) Colormap

Figure 8: Population data from U.S. Census Bureau on various levels for the year 2000 – The area of each map partition corresponds to the number of people living there. The colormap of Figure 8(h) indicates the number of people living in each region (yellow: high population; brown: low population) and is a link from the traditional map to the corresponding map partitions.