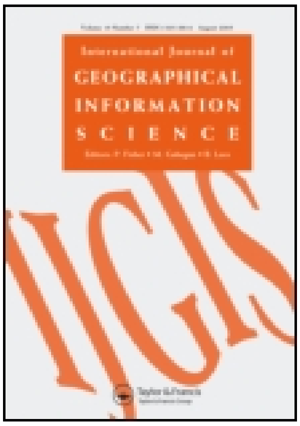


International Journal of Geographical Information Science
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tgis20

Carto-SOM: cartogram creation using self-organizing maps

R. Henriques [a] , F. BaÇão [a] & V. Lobo [a b]

[a] Institute of Statistics and Information Management, New University of Lisbon (ISEGI-UNL), Campus de Campolide, 1070-312 Lisboa, Portugal

[b] Portuguese Naval Academy, Alfeite, 2810-001 Almada, Portugal

Available online: 08 Jun 2009

To cite this article: R. Henriques, F. BaÇão & V. Lobo (2009): Carto-SOM: cartogram creation using self-organizing maps, International Journal of Geographical Information Science, 23:4, 483-511

To link to this article: http://dx.doi.org/10.1080/13658810801958885

International Journal of Geographical Information Science
Vol. 23, No. 4, April 2009, 483–511

Research Article

Carto-SOM: cartogram creation using self-organizing maps

R. HENRIQUES*†, F. BAÇÃO† and V. LOBO†‡
†Institute of Statistics and Information Management, New University of Lisbon (ISEGI-UNL), Campus de Campolide, 1070-312 Lisboa, Portugal
‡Portuguese Naval Academy, Alfeite, 2810-001 Almada, Portugal

(*Received 16 March 2007; in final form 1 February 2008*)

The basic idea of a cartogram is to distort a geographical map by substituting the geographic area of a region by some other variable of interest. The objective is to rescale each region according to the value of the variable of interest while keeping the map, as much as possible, recognizable. There are several algorithms for building cartograms. None of these methods has proved to be universally better than any other, since the trade-offs made to get the correct distortion vary. In this paper we present a new method for building cartograms, based on self-organizing neural networks (Kohonen's self-organizing maps or SOM). The proposed method is widely available and is easy to carry out, and yet has several appealing properties, such as easy parallelization, making up a good tool for geographic data presentation and analysis. We present a series of tests on different problems, comparing the new algorithm with existing ones. We conclude that it is competitive and, in some circumstances, can perform better then existing algorithms.

Keywords: Cartograms; Neural networks; Kohonen self-organizing maps

1. Introduction

Maps have always been an important part of human life, forming one of the oldest means of human communication. Today, maps continue to play an important role in our society describing and communicating many geographic (and other) facts. They may be regarded as a means for converting large amounts of data into information that can be easily understood by human beings. The arrival of personal computers and the consequent increase in geoprocessing capacities had an impact not only on map availability but also on map-making tools. Today almost everybody has access to tools which make maps at the push of a button.

Cartograms are a type of map or a map variant. The difference between most traditional maps and a cartogram is the variable that defines the size of the regions. In most traditional maps this variable is the geographic area of the regions (or a good approximation of it), while cartograms may use any other georeferenced variable. In the context of the famous framework provided by the seven visual variables of Bertin (1983), the cartogram uses the visual variable size to encode a data variable.

In this paper we present a new algorithm to create cartograms based on a self-organizing neural network, proposed by Kohonen (2001).

*Corresponding author. Email: roberto@isegi.unl.pt

This paper begins with the definition of cartograms and reviews the most popular algorithms for building cartograms. We then present a thorough description of the new algorithm based on the use of self-organizing maps (Carto-SOM). To test the Carto-SOM method we use three different datasets, and two other algorithms to build cartograms are used for benchmarking. Finally we perform a quantitative evaluation of the cartograms produced by the Carto-SOM and the benchmark algorithms.

2. Cartograms

According to Tobler (2004) there are two generic types of cartograms. The first form of cartogram distorts the space according to a non-geographic metric, such as cost or time (Tobler 1970). The second form of cartogram distorts the space according to a georeferenced variable. Depending on the cartogram regions' continuity cartograms are classified as non-continuous (Olson 1976) or as continuous. In this paper we will focus only on continuous cartograms, and for simplicity will use the word cartogram to refer to these.

'Cartograms are purposely distorted maps that highlight a variable distribution by changing the area of objects on the map' (Changming and Lin 1999). *'Area cartograms are deliberate exaggerations of a map according to some external geography-related variable that communicates information about regions through their spatial dimensions'* (Dougenik *et al.* 1985). *'These dimensions have no correspondence to the real world but are a representation of one variable other than the area'* (Kocmoud 1997).

In a population cartogram, for example, the sizes of regions are proportional to the number of inhabitants. While population is the most commonly used, we can use any social, economic, demographic or geographical variable to build a cartogram.

2.1 *Problem definition*

As Keim *et al.* (2005) point out cartogram generation is a map deformation problem. The inputs of the problem are:

- a polygon map composed of a set of regions, each with an initial area given by the true geographical area (each polygon matches a region);
- a target value for the final area of each one of the polygons (regions), representing the variable of interest in that region.

The goal of the map distortion is to approximate, as much as possible, the areas of the regions to the desired target.

The cartogram problem may be formally described as follows:

Let $M=\{R_0, R_1, \ldots R_n\}$ be a map M consisting on n regions R_i, each of which forms a polygon.
Let $T(M)$ be the contiguity matrix of M.
Let each region R_i have an area of a_i (area of its polygon) and a 'value of interest' v_i (usually population, average income, or some other variable).
Produce a cartogram map $C=\{R'_0, R'_1, \ldots R'_n\}$ consisting of n regions R'_i, each of which forms a polygon with area $a'_i=kv_i$, with $k=\Sigma a/\Sigma v$, and $T(C)=T(M)$.

There is still another goal, loosely described as 'shape similarity'. Let S be the shape of M and S' the shape of C. The objective is that $S \cong S'$. The 'shape similarity' is an elusive concept that translates the ability of a reader to recognize C as an instance of M. This property is difficult to measure and difficult to define rigorously. The ability to recognize C is not only dependent on the ability to preserve the shape of each R_i but also on preserving certain landmark points. For instance recognizing a certain cartogram as a cartogram of the United States is much more dependent on preserving the shape of Florida than on preserving the shape of North Dakota.

There are many possible cartograms C that achieve the desired goal, but most mappings from M to C are the result of an iterative process, and only asymptotically get the desired result (or at least some approximation to it).

3. Methods for building cartograms

Several computer algorithms have been developed to build continuous area cartograms, and in this paper we will briefly review some of the most important ones. The reader interested in an extensive and thorough review of the work done in the last 35 years in cartograms is referred to a recently published paper by Tobler (2004). In this paper we briefly review eight algorithms that represent different approaches to the problem and are the most important and effective ones: *Rubber-map cartogram* (Tobler 1973), *Contiguous Area Cartogram* (Dougenik *et al.* 1985), *Pseudo-cartogram* (Tobler 1986), *Line Integral Method* (Gusein-Zade and Tikunov 1993), *Constraint-based Continuous Area Cartogram* (House and Kocmoud 1998), *Medial-axes-based cartogram* (Keim *et al.* 2002, 2004, 2005), *Rectangular Map Approximations* (Heilmann 2004) and *Diffusion cartograms* (Gastner and Newman 2004).

The *Rubber-map cartogram* (Tobler 1973) was one of the first methods created for population cartograms. This method distorts a map as if it were a rubber surface. Tobler's method initially divides the map into a regular grid, computing a density value for each grid cell. On each cell vertex a displacement direction is computed to minimize the 'density error' of its four adjacent cells. This displacement continues until no improvement can be made. Since there is a one-to-one mapping between the initial and final grids a double bivariate interpolation is used to project the map vertices to and from the final grid.

One of the most famous algorithms to build continuous cartograms is *Dougenik's Contiguous Area Cartogram* (Dougenik *et al.* 1985). This algorithm is based on Tobler's rubber map and uses a model where forces are exerted from each polygon centroid, acting on its boundary. In this method each polygon centroid applies a force F to its vertices. This force will move vertices away if F is a positive value and will bring them closer if F is negative. The intensity of the force exerted on each polygon is determined by the difference between the *current* and the *desired* polygon area. The *current area* is the area of the polygon in each iteration, while the *desired area* is the distorted area in the final cartogram. In this method, some overlapping might, occasionally, occur due to complexity of the polygon shapes. However, it presents an improvement in speed compared to Tobler's method (Tobler 2004). A 2001 population cartogram, built with this algorithm, of the USA is shown in Figure 1.

Another cartogram construction algorithm, introduced in 1986 by Tobler, is the *pseudo-cartogram* (Tobler 1986). This method is similar to the rubber-map method and seeks to reduce the area error by adjusting the lines of latitude and longitude on the map. Tobler's pseudo-cartogram algorithm starts with a grid superimposed on the map, after which the horizontal and vertical lines of the grid move, compressing or expanding the map to get equal density estimation. As with the rubber-map method, this algorithm suffers from a high dependency on the coordinate system.

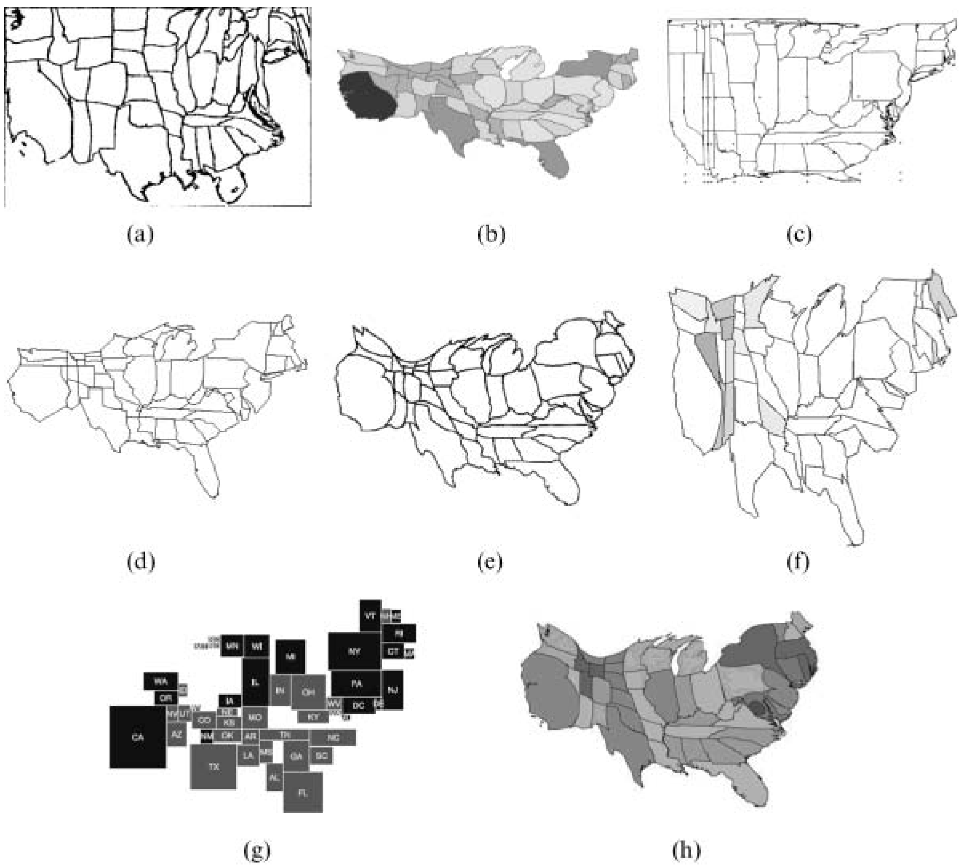


Figure 1. Cartograms of USA population by state, using different cartogram building algorithms. (a) Rubber-map cartogram (Tobler 1973); (b) Contiguous area cartogram (Dougenik *et al.* 1985); (c) Pseudo-cartogram (Tobler 1986); (d) Constraint-based Continuous area cartogram (House and Kocmoud 1998); (e) Line Integral Method (Gusein-Zade and Tikunov 1993); (f) Medial-axes-based cartogram (Keim *et al.* 2002, 2004, 2005); (g) Rectangular map approximations (Heilmann *et al.* 2004); (h) Diffusion cartograms (Gastner and Newman 2004).

Zade and Tikunov proposed the *Line Integral Method* which applies radial transformations such that the density of the final map is uniform (Gusein-Zade and Tikunov 1993). The algorithm initially divides the map into a grid of cells. Radial transformations are applied to each cell making its density uniform, and changing the shape of the overall map.

In 1998 House and Kocmoud (House and Kocmoud 1998) proposed the *Constraint-based Continuous Area Cartogram*. According to this method cartogram creation is seen as a constrained optimization problem. The map is a connected collection of polygons and an *area function* decides each region's desired area, according to the total map area and variable of interest. The goal, as in any cartogram, is to rescale each object keeping map recognition. The search space is composed by all the maps that keep topology and respect the area function values of each region. Within these possible solutions, the best map, which is the most recognizable one, is selected using optimization heuristics.

The *medial-axes-based cartogram* (Keim *et al.* 2002, 2004, 2005) uses the medial axes of an object to build a cartogram. The medial axis is a simplification of the shape of an object and can be obtained by connecting the centers of the maximum circles inscribed in the object. The regions are then expanded or compressed along lines perpendicular to the medial axis (cutting lines). This method presents as a major advantage its processing time, which the authors claim to be several times lower than previous methods.

Another cartogram algorithm is the *Rectangular Map Approximation* (Heilmann *et al.* 2004), which represents each region as a rectangle. Each rectangle area is proportional to some variable. To best recognize the cartogram information, the method places rectangles as much as possible in the original position and as close as possible to their neighbors. This method uses genetic algorithms to select the best cartogram using a given cost function which is defined using the region's area, shape, topology, relative position and the spaces without any region. Although this method produces cartograms with low area errors, it presents two major drawbacks: topology error is never null, *i.e.* there will always be some topological differences between the original and the cartograms regions, and cartogram readability is hampered due to the use of rectangles instead of the original shapes.

Finally, the *diffusion cartograms method* (Gastner and Newman 2004) produces cartograms where density variations cause diffusion from high-density areas into low-density ones. This method uses the linear process of diffusion from elementary physics to achieve a solution. It is a simple method, providing a tool to quickly calculate accurate cartograms. By changing the way that density is estimated, it also allows the user to switch between more detailed or more easily readable cartograms, thus increasing its applicability. One of its main advantages is that, using a continuous problem formulation, it can achieve accurate cartograms even for a high number of regions (more than 3000). On the other hand it can be argued that cartograms with a large number of features (*i.e.* 3000 regions) are not as useful as smaller ones, due to the difficulty in recognizing the original regions.

From this brief review of the work done in developing algorithms for cartogram building there are three major conclusions that can be drawn: first, none of the existing methods is universally better than any other; second, most of these algorithms are computationally demanding; third, the potential user will encounter difficulties and challenges in the implementations of the algorithms. These conclusions suggest that one should examine the different options as they vary in computational efficiency, ease of implementation and performance. Thus, a new method that produces different and competitive cartograms can be a useful addition to the set of algorithms already available.

3.1 *Quantitative evaluation of cartograms*

While it is subjective to compare the visual quality of different cartograms, it is easy to define a numerical value that characterizes how well the cartogram shares the available area between the different regions, according to the given variable of interest. Various such measures have been proposed. Keim (2004) proposes an area error function to determine the cartogram error in each region. This relative area error e^{i}_{rel} of a region R_i is given by:

$$e^{i}_{rel} = \frac{\left|a'_i - a_i^{current}\right|}{a'_i + a_i^{current}}$$

where a'_i is the desired area of region R_i in the cartogram (to create a perfect cartogram) and $a^{current}{}_i$ is the area of region R_i in the cartogram.

Other possible error measures could be used, such as the one proposed by Kocmoud (1997), or by Henriques (2006), but while having specific advantages and disadvantages, they do not produce significantly different results (Henriques 2006).

The error used in this paper was Keim's, and may assume values in [0,1]. For a perfect cartogram, the error is zero, since in that case each region occupies an area strictly proportional to the value of its variable of interest. The extreme but unattainable value 1 would mean that all the map space is used up by a region where the value of interest is zero, or *vice versa*.

3.2 *Global cartogram error*

The error measure given previously is defined for each individual region. Thus, we need to define a global error that may characterize the overall quality of the cartogram.

Based on the error $e(i,v)$ of each region R_i in regard to the variable of interest v, we may define a global measure of the 'goodness' of a cartogram by its weighed mean, simple mean or quadratic mean. The quadratic mean error (*mqe*) is defined as:

$$mqe(v) = \frac{\sqrt{\sum_{i=1}^{N} e(i, v)^2}}{N}$$

The weighed mean error (*we*) is:

$$we(v) = \sum_{i=1}^{N} |a_i \times e(i, v)|$$

The simple average error (*se*) is:

$$se(v) = \sum_{i=1}^{N} |e(i, v)|$$

These error measures may be applied both to a cartogram and an ordinary geographical map. The global error of a geographical map is a measure of how much the variable of interest is misrepresented by the geographical area of the different regions, and thus how much distortion is necessary to get an accurate cartogram.

4. A new approach for building cartograms based on self-organizing maps (SOM)

In this paper we present a new algorithm to build cartograms based on the SOM. Teuvo Kohonen proposed the SOM in the beginning of the 1980s (Kohonen 1982) as a result of his work on associative memory and vector quantization. Although the term 'self-organizing map' could be applied to a number of different approaches, we shall use it as a synonym of Kohonen's Self Organizing Map, or SOM for short. These maps are also referred to as 'Kohonen Neural Networks' (Fu 1994), 'Self-Organizing Feature Maps – SOFM', 'topology preserving feature maps' (Kohonen 1995), or some variant of these names.

One of SOM's main objectives is to '*extract and show*' the essential structures in a dataset, through a map resulting from an unsupervised learning process (Kaski and

Lagus 1996, Kaski and Kohonen 1998). The SOM is normally used as a tool for mapping high-dimensional data into one, two, or three-dimensional feature maps. The main advantage of the SOM is that it allows us to have some idea of the structure of the data through the analysis of the resulting map. This is possible mainly because the SOM preserves topological relations.

The basic idea of a SOM is to map the data patterns onto an *n*-dimensional grid of units or neurons. That grid forms what is known as the output space, as opposed to the input space that is the original space of the data patterns. This mapping tries to preserve topological relations, *i.e.* patterns that are close in the input space will be mapped to units that are close in the output space, and *vice versa*. The output space will usually be two-dimensional, and most of the implementations of SOM use a rectangular grid of units. So as to provide even distances between the units in the output space, hexagonal grids are sometimes used (Kohonen *et al.* 1995). Single-dimensional SOM's are also common, *e.g.* for solving the traveling salesman problem (Maenou *et al.* 1997), and some authors have used three-dimensional SOM's (Takatsuka 2001). Using higher dimensional SOM's (Villmann *et al.* 2003), although posing no theoretical obstacle is rare, since it is not possible to easily visualize the output space.

Each unit, being an input layer unit, has as many weights (sometimes also referred to as coefficients) as the input patterns, and can thus be regarded as a vector in the same space of the patterns. When training a SOM with a given input pattern, the distance between that pattern and every unit in the network is calculated. Then the algorithm selects the unit that is closest as the winning unit (also known as best matching unit), and that pattern is mapped onto that unit. If the SOM has been trained successfully, then patterns that are close in the input space will be mapped to neurons that are close (or the same) in the output space. Thus, SOM is 'topology preserving' in the sense that (as far as possible) neighborhoods are preserved through the mapping process.

Before training, the units may be initialized randomly (Kohonen 2001). Typically, the training is done in two parts. During the first part of training, the units are 'spread out', and pulled towards the general area (in the input space) where they will stay. This is usually called the unfolding phase of training. After this phase, the general shape of the network in the input space is defined, and we can then proceed to the fine tuning phase, where we will match the units as best as possible to the input patterns, thus improving the quality of the mapping provided by the network. Due to the way we use and initialize the SOM in our application, the first phase (unfolding) may be omitted.

The quality of the mapping is usually assessed through the use of the quantization error (Kohonen 2001), which gives the user information on how well the SOM fits the data. The quantization error is calculated by measuring the distance between each input pattern and the closest unit, consequently giving a measure of the distance between the units and the data they are suppose to represent. Thus a low average quantization error constitutes a criterion for a good result.

The basic SOM learning algorithm may be described as follows:

```
Let
X be the set of $n$ training patterns $x_1, x_2, \ldots x_n$
W be a $p \times q$ grid of units $w_{ij}$ where $i$ and $j$ are their coordinates on that grid
$\alpha$ be the learning rate, assuming values in [0,1], initialized to a given
initial learning rate
$r$ be the radius of the neighborhood function $h(w_{ij}, w_{mn}, r)$, initialized
to a given initial radius
```

```
1. Repeat
2.   For k=1 to $n$
3.       For all $w_{ij} \in$ W, calculate $d_{ij}=\|x_k-w_{ij}\|$
4.           Select the unit that minimizes $d_{ij}$ as the winner $w_{winner}$
5.           Update each unit $w_{ij} \in$ W: $w_{ij}=w_{ij}+\alpha h(w_{winner}, w_{ij}, r)\,\|x_k-w_{ij}\|$
6.   Decrease the value of $\alpha$ and $r$
7. Until $\alpha$ reaches 0
```

The learning rate α, sometimes referred to as η, varies in [0,1] and must converge to zero so as to guarantee convergence and stability in the training process. The decrease from the initial value of this parameter to zero is usually done linearly, but any other function may be used.

The radius, usually denoted by r, indicates the size of the neighborhood around the winner unit in which the units will be updated. This parameter is particularly relevant in defining the topology of the SOM, deeply affecting the unfolding of the output space. Initially r can be as large as the size of the network but, in order to guarantee convergence and stability, it has to converge to 1 or 0. For the sake of simplicity, r is sometimes omitted as an explicit parameter of the neighborhood function. The update of both α and r may be done after each training pattern is processed (commonly used in online training) or after the whole training set is processed (commonly used in batch training).

The neighborhood function h, sometimes referred to as Λ or N_c, assumes values in [0,1], and is a function of the position of two units (a winner unit, and another unit), and radius r. It is large for units that are close in the output space, and small (or zero) for units far away. Usually, it is a function that has a maximum at the centre, monotonically decreases up to a radius r and is zero from there onwards. The two most common neighborhood functions are the bell-shaped (Gaussian-like) and the square (or bubble).

Like in all heuristic methods the parameters that control the training have an important role in the quality of the SOM final results. Different parameters will lead to different results, and that is why it is usual practice to perform several runs and choose the one with the minimum quantization error (Kohonen 2001).

4.1 *Building cartograms using SOM*

The idea of using an SOM in cartography has already been explored by Skupin (2003) who uses it as a nonlinear projection tool. Our idea for using SOM to build cartograms stems from the fact that it can be seen as a density estimation tool (Sarajedini and Chau 1996, Kohonen 2001, Merenyi *et al.* 2007).

The properties of the SOM as a density estimation tool have not yet been completely established, except for some very particular cases, *e.g.* the one dimensional case (Ritter and Schulten 1988, Cottrell *et al.* 1998). In these cases it has been found that the SOM has a bias towards low density areas, *i.e.* $D_{units}=K^{*}D_{data}{}^{\mu}$, where K is a constant scaling factor, and μ, known as magnification factor, is 2/3 for some known cases. A magnification effect of 1 would mean strict proportionality between the density of input patterns and the density of their assigned units. With a magnification factor of less than 1 low density areas will be proportionally over represented. Some work has been done to calculate the magnification factor in more general cases (Dersch and Tavan 1996, Fort 2006, Merenyi *et al.* 2007), and experimental evidence suggests that when using the original

SOM algorithm the magnification factor is always less than one. Some methods have been proposed to explicitly control the magnification factor (Bauer *et al.* 1996, Merenyi *et al.* 2007), but in this paper we will use the original SOM algorithm.

In spite of the magnification effect, the more input patterns exist in a specific area of the input space the more units are assigned to represent that specific area. Additionally, units that represent that area are neighbors in the output space, consequently they can be labeled with the same label. Thus, if we populate a particular map with random points, with a density distribution proportional to the value of the variable of interest, we can train the SOM with that data. We can then label the units according to the points they are representing and use the output space as a cartogram. The process can be seen as the folding of a blank sheet (in this case according to the density of the points that represent the variable of interest) painting it according to a reference map and finally spreading out the sheet completely thus obtaining the cartogram. This approach is particularly interesting because we do not alter the standard training procedure in any way and thus any SOM implementation may be used to produce cartograms.

In our proposal the cartogram is built based on the unit's movement during the learning process of the SOM (Henriques 2006). In essence, during that learning process, units are moved towards higher density areas. If that movement is recorded, its inverse will transform the original map into an equal density map.

Figure 2 shows an example of this process, using a simplified instance with only two regions. The two regions are geographically identical but have distinct values for the variable of interest, represented by p. The variable p has a high value in the dark region (*i.e.* this is a region with a high population density), and a smaller value in the lighter region. In the following step (Figure 2(b)), we randomly produce points with a uniform distribution (represented with triangles) inside each region. The number of points created is a linear function of the value of population p. Figure 2(c) shows an initialized two-dimensional SOM (with 4×4 units). The SOM units are initialized so as to form a regular grid in the input space, contrary to the usual practice, in which the units are randomly initialized in the input space. We then continue with a standard SOM training phase where the units adapt to the training patterns. This means that units are moved (in the input space) in such a way that their density mimics the density of the data patterns (Figure 2(d)). The algorithm continues with the labeling process (Figure 2(e)). This process gives each unit a label based on the data of the region where it lies. In this case, the labels are the colors that identify each region, and thus each unit will be assigned the color of the region where it lies after training. Figure 2(f) represents each unit mapped back at its initial position. Based on the units' positions and labels a final population cartogram is produced (Figure 2(h)). In this cartogram the darker region (that had more population) is larger than the lighter one (that had fewer population), and thus the final area of the regions is proportional to their population.

Most SOM implementations use rectangular shaped maps such as the one in this example. Generally, the geographic area of interest is not rectangular, and forcing the final cartogram to be rectangular, as happens with the described method, is not acceptable. The root of this problem is that when the SOM is initialized with a regular grid in a rectangle, some of its units will fall outside the geographic area of interest, as shown in Figure 3(a).

During training those units will move to areas populated with training patterns (Figure 3(b)). When comparing the produced cartogram (Figure 3(c)) with the original

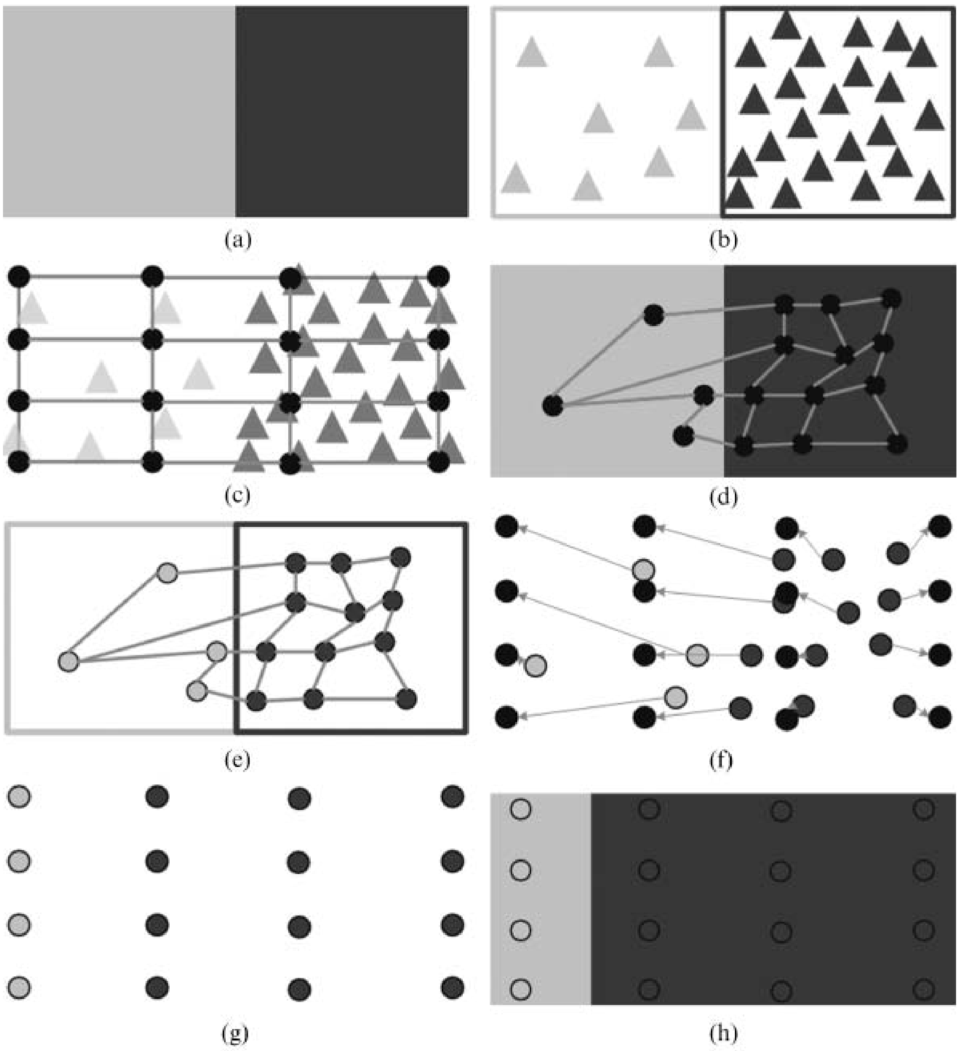


Figure 2. Proposed method example.

regions shape (Figure 3(a)) it is apparent that the cartogram regions will occupy all the (rectangular) area on the input space losing the original regions' shapes.

To address this problem we define three different areas in the input space (Figure 4). The input space is the domain area (*D*); the area occupied by the original regions is the region area (*ra*) and the remaining area is the buffer area (δ).

The input space can now be defined as:

$$D = ra + \delta$$

As already shown, when no training patterns lie in the buffer area (δ) the cartogram produced will occupy the whole domain area (*D*) losing the original shape of the region area (*ra*). To solve this problem we created and tested several different solutions presented in detail in Henriques (2006). Among these, the solution that produced the best cartograms consists in generating data points within the buffer area (δ) with a uniform distribution, and a density equal to the average density of data points in the

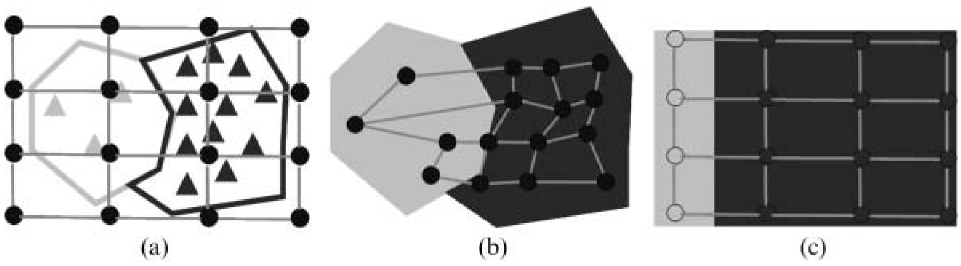


Figure 3. Rectangular shaped SOM superposed on a non-regular shaped dataset. (a) Random point creation based on a region feature. (b) SOM units after training. (c) Produced cartogram.

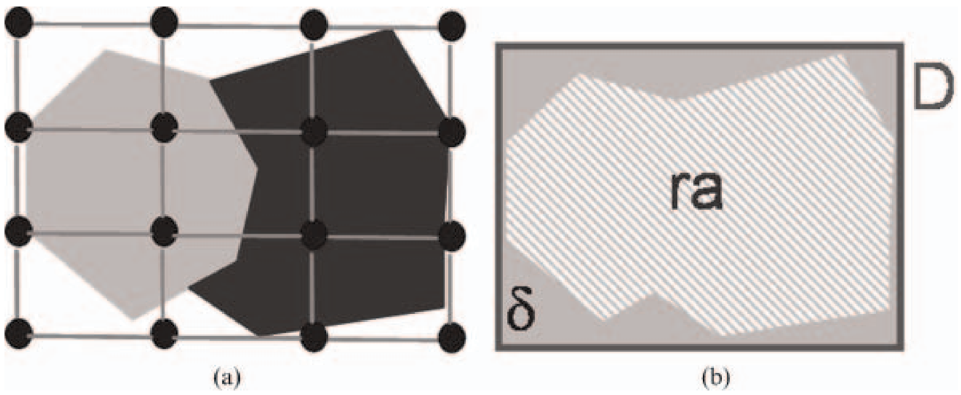


Figure 4. Input space nomenclature. (a) SOM mapped in the input space. (b) Input space area definition: *ra* is the region area, and δ is the buffer area.

whole region of interest (*ra*). In fact other authors (Tobler 1973, Gastner and Newman 2004) also used a similar technique to obtain good cartograms.

4.2 *Compensating for the magnification effect of SOM*

Although the proposed algorithm may use the data available without any preprocessing, the magnification effect of the SOM, described earlier, will distort the results, producing cartograms where low density areas will be proportionally overrepresented. In some cases, this may actually be desirable, so as to prevent low density areas from 'disappearing'. Nevertheless, we may correct, at least in part, this effect by *boosting* the original data, so that low density areas will have even lower *surrogate* densities to compensate for the magnification. We can then use this *surrogate* dataset instead of the original one. Let us now see in detail how this dataset may be generated.

For a one-dimensional to one-dimensional mapping, under the special conditions described by Ritter (1991), the relationship between densities of units and data points is given by:

$$D_{units} = k_1 D_{data}^{\mu}$$

where D_{units} is the density of SOM units, D_{data} is the density of data points, k_1 is a proportionality constant that results from the ratio between the total number of units and the total number of data points, and μ is the magnification factor.

To build a good cartogram, we want strict proportionality between the density of units and the density of the original data. To obtain this we must have a surrogate dataset where the density D_{surr} is such that

$$D_{units} = k_1 D_{surr}^{\mu} = k_2 D_{data} \Leftrightarrow D_{surr} = \left(\frac{k_2}{k_1}\right)^{\frac{1}{\mu}} D_{data}^{\frac{1}{\mu}} = k_3 D_{data}^{\frac{1}{\mu}}$$

where k_3 is $(k_2/k_1)^{1/\mu}$. For the sake of simplicity we will not explicitly calculate each of the constants k_x involved in these calculations, since in the end we will be able to calculate the necessary variables without them.

Since we will be generating, for each region, a number of points proportional to the variable of interest V and not its density, we have

$$D_{surr} = k_3 D_{data}^{\frac{1}{\mu}} \Leftrightarrow \frac{V_{surr}}{A} = k_3 \frac{V_{data}^{\frac{1}{\mu}}}{A^{\frac{1}{\mu}}} \Leftrightarrow V_{surr} = k_3 \frac{A}{A^{\frac{1}{\mu}}} V_{data}^{\frac{1}{\mu}} = k_3 A^{1-\frac{1}{\mu}} V_{data}^{\frac{1}{\mu}}$$

where V_{surr} is the surrogate variable of interest to be used instead of the variable of interest V_{data} for each region with area A. The number N of data points generated for each region is proportional to the variable of interest, *i.e.* $N = k_4 V_{surr}$. Combining this with the previous equation we obtain, for each region,

$$N = k_5 A^{1-\frac{1}{\mu}} V_{data}^{\frac{1}{\mu}}$$

To compute the constant k_5 we only need to know how many data points we want overall N_{total}, since

$$N_{total} = \sum_{all\,\mathrm{Re}\,gions} k_5 A^{1-\frac{1}{\mu}} V_{data}^{\frac{1}{\mu}} \Leftrightarrow k_5 = \frac{N_{total}}{\sum\limits_{all\,\mathrm{Re}\,gions} A^{1-\frac{1}{\mu}} V_{data}^{\frac{1}{\mu}}}$$

If the original expression is valid for two-dimensional to two-dimensional mappings, and we knew the magnification factor exactly, then this correction would allow a perfectly proportional representation of each region, and thus a null error in the cartogram. Since that expression is just an approximation, the error will in fact be greater than 0. As for the magnification factor μ, it is reasonable, from empirical experience, to assume it is approximately 2/3 (Merenyi *et al.* 2007). The final number of points for each region i will thus be given by

$$N_i = k A_i^{1/3} V_i^{3/2} \text{ and } k = \frac{N_{total}}{\sum\limits_i A_i^{1/3} V_i^{3/2}}$$

Since the area outside the region of interest (δ) does not have to be faithfully represented, this correction need not be applied to that region.

4.3 *Carto-SOM algorithm*

The Carto-SOM algorithm may be defined as follows:

Given a dataset consisting of geographic map M with n regions R_i each with a geographical area a_i and a 'value of interest' v_i, do the following:

1. Define a rectangle P encompassing all regions R_i.

2. Define a new region $R_{n+1}=\delta$ consisting of all areas within P not covered by any R_i, and assign it a value of interest $v_{i+1}=\Sigma v_i/\Sigma a_i \times a_{i+1}$, where the sums are over all other regions. This corresponds to assigning a density for the *value of interest* of the area outside the original map equal to the average of the rest of the map.
3. For each region, generate $m=k \times a_i^{1/3} v_i^{3/2}$ data points with coordinates within that region, and assign to them a label i that identifies them as belonging to region R_i. The constant k is equal for all regions and it determines the total number of data points used. The choice of this number depends on the scale of the value of interest and the desired quality of the cartograms. Large values of k will produce smoother cartograms, but will require more processing power. In practice, k should be chosen so that even the smallest region gets at least 10 points, *i.e.* given the region j with smallest value of interest and smallest area, $k \geqslant 10/(a_j^{1/3} v_j^{3/2})$. The generated points may have a random uniform distribution within the region, or may be evenly spaced within that region. Optionally, this third step may be simplified by generating $m=k \times v_i$ data points within each region. This last option assumes that the magnification is 1, and thus leads to a slightly larger cartogram error, as we shall see later.
4. Generate an initial SOM with $x \times y$ units evenly spaced in the geographic rectangle P. The ratio x/y should be the same as ratio of width to height of rectangle P, and the larger the value of x the smoother the cartogram will be. In practice, the value of x should be as high as possible, but taking into account the processing power available. Depending on the complexity of the shapes a value of 100 will usually be enough.
5. Train the SOM with the data points obtained in step 3, using the standard algorithm and 'standard' training parameters (in section 5.2. we study more closely the effects of different training parameters).
6. Label the SOM with the labels of the data points that are mapped to it. It is possible to do this with the standard SOM labeling routine available in most SOM software packages. In some rather awkward situations this may produce a few unlabeled units. While not critical, these units may be labeled using the reverse labeling process described in the work by Henriques (2006), which basically consists in assigning to each SOM unit the label of the closest data pattern.
7. Print out the labels of each unit, using as coordinates each unit's position in the SOM (*i.e.* in the output space). This is the desired cartogram.

Steps 1 to 3 are pre-processing steps that can be performed using any geographic information system, such as ArcGIS or MapInfo, purpose built Matlab routines (available at our site), or even a general-purpose program such as Excel.

Steps 4 to 7 may be performed with any SOM software, such as SOMTOOLBOX for Matlab, SOM-PAK, or SAS Enterprise Miner. However, to obtain the best results and avoid possible problems with unlabeled units, the routines available for Matlab at http://www.isegi.unl.pt/labnt/roberto/ should be used.

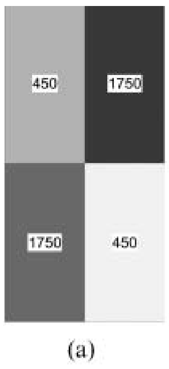
5. Results

In this section we test the Carto-SOM method using three different datasets. An analysis of Carto-SOM's sensitivity to training parameters is performed along with a comparison with Dougenik and Gastner's cartograms.

5.1 *Experimental settings*

To test the Carto-SOM method we used an artificial dataset, Portugal's population data for 2001 and USA's population data for 2000 (Figure 5).

We chose the Dougenik cartogram method (Dougenik *et al.* 1985) and the diffusion method (Gastner and Newman 2004) as benchmarks for the Carto-SOM.



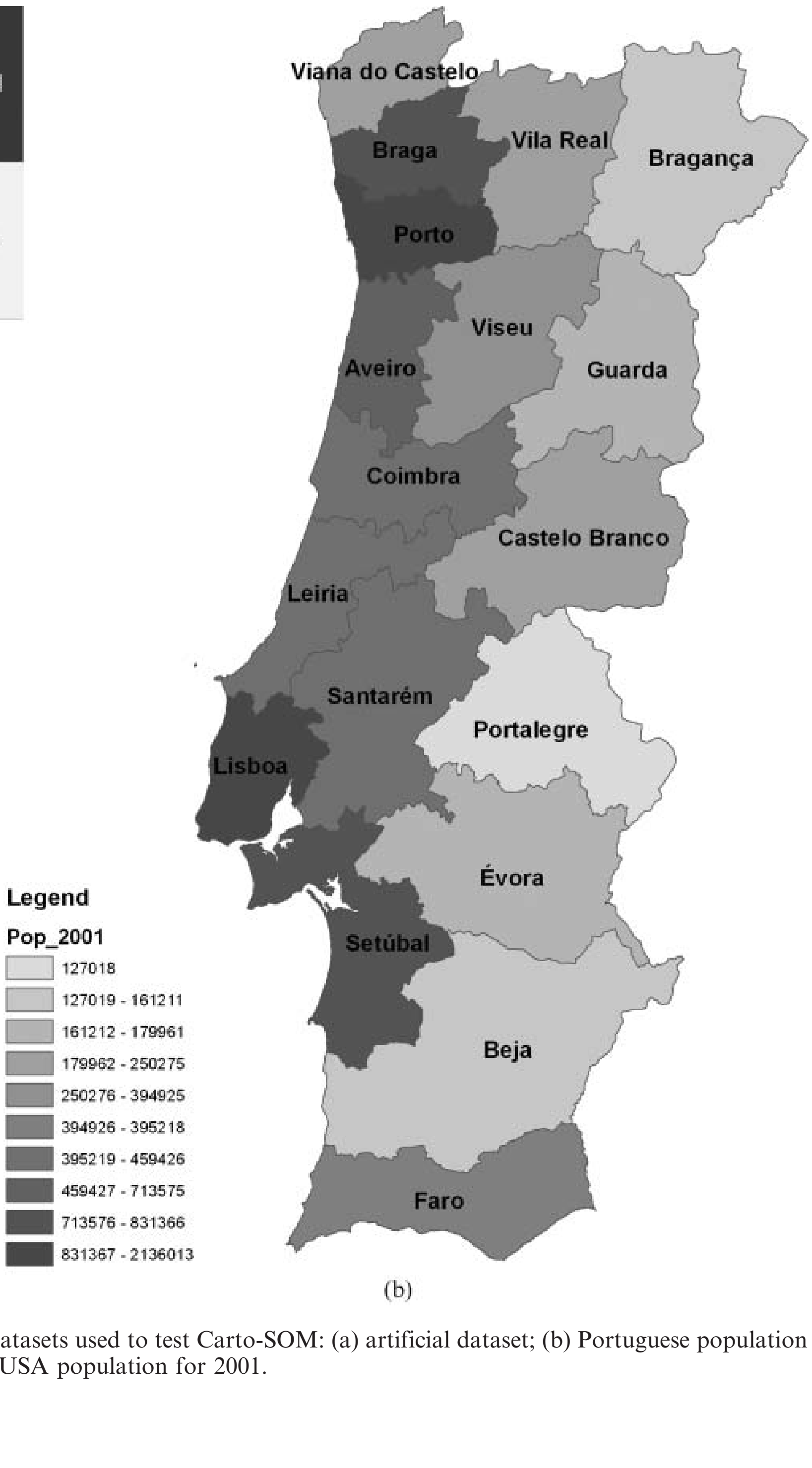


Figure 5. Datasets used to test Carto-SOM: (a) artificial dataset; (b) Portuguese population for 2001; (c) USA population for 2001.

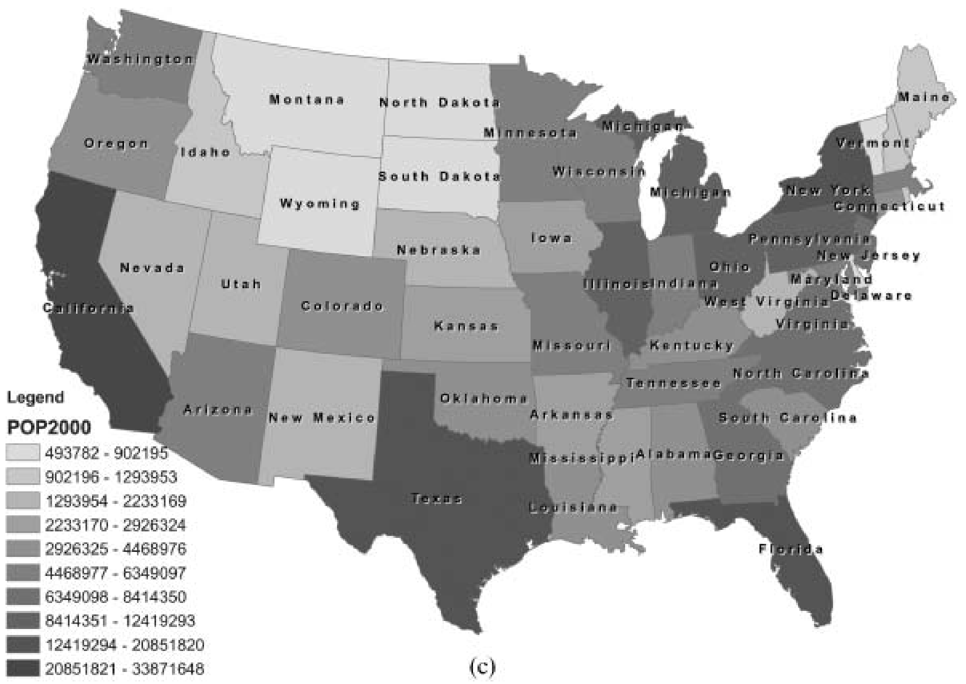


Figure 5. (*Continued.*)

The choice of Dougenik's cartograms was due to the fact that these are quite popular and the software (to implement them in an ArcGIS environment) is freely available at http://arcscripts.esri.com/details.asp?dbid=14226. The choice of diffusion cartograms was due to the fact that they seem to be the ones that manage to obtain the best quantitative errors when representing the variable of interest.

To test the Carto-SOM method we trained the SOM using the MATLAB SOM-Toolbox (Vesanto *et al.* 2000). Several training processes were performed, changing the initial SOM parameters. In Table 1 we present the parameters that provided the best cartograms. In the next section we present the results of the sensitivity analysis performed based on the USA population dataset.

For Dougenik's method, we used an ArcGIS script file to produce the cartograms (Goldschmidt and Passos 2005). This script allows the ArcGIS user to select an input layer and to choose the number of iterations used. In this test we used five iterations to produce Dougenik's cartograms.

For the diffusion cartogram we used an implementation produced by Michael Gastner available at his homepage (Gastner 2005).

Table 1. B SOM parameters used as reference in the sensitivity analysis with the USA dataset.*

Row	Column	r	α	E
258	388	26	0.1	40

**Row* and *column* are the number of *rows* and *columns* used in the SOM; r is the neighbourhood radius, α is the learning rate, E is the number of epochs used in the training phase.

5.2 *Sensitivity analysis of Carto-SOM*

In order to analyze the sensitivity of the Carto-SOM method we performed several sets of tests changing in each set one of the parameters. In the first set we test Carto-SOM's robustness to initialization and random effects by using the same parameters in several tests. In the second set we assumed different magnification factors when boosting the original data, to conclude about the effect of SOM's magnification problem in the cartograms. In the third set of tests, which we call the SOM parameters evaluation, we test changes in the neighborhood radius, learning rate and the number of epochs. Finally in the fourth set we evaluate influence of the number of units of the SOM and input data points.

5.2.1 Carto-SOM robustness test. To evaluate the influence of random effects due to the particular sample of input data points chosen, and the order by which they are selected and presented to the algorithm, we performed 10 tests. In these tests all Carto-SOM parameters were maintained except the input data points which were generated each time (Table 2).

In Figure 6 we present the Keim simple error for each cartogram produced. For the 10 tests performed we achieved an average error of 4.36% with a standard deviation of 0.32.

From these results we may conclude that Carto-SOM is quite robust to different samples of input data points. These tests prove that generating input data points in a randomly way will not affect the final cartogram provided that the density is maintained. They also prove that even though Carto-SOM uses a neural network that may produce slightly different results each time it is used, the differences are negligible.

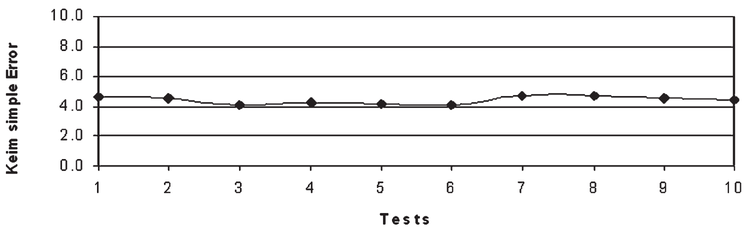
5.2.2 Magnification effect. In order to evaluate the magnification effect (used when boosting the original data), we performed several different tests changing the magnification factor value as shown in Table 3. Test 11, where $\mu=1$, corresponds to a proportional input data point generation, *i.e.* there is no compensation for the magnification effect.

In Figure 7 the Keim simple error for each of the eleven cartograms produced is presented. The lowest error is obtained when the value of μ is 0.7 (similar to the

Table 2. SOM parameters used to test input data points influence.*

Row	Column	r	α	E	nD
258	388	26	0.1	40	20 000

**Row* and *column* are the number of *rows* and *columns* used in the SOM; r is the neighborhood radius, α is the learning rate, E is the number of epochs used in the training phase and nD is the number of input data points used which were randomly generated for each test.



Keim simple error	
mean	St. deviation
4.36	0.32

Figure 6. Carto-SOM robustness to the choice of input data points.

Table 3. Magnification factor (μ) variation.*

Test	1	2	3	4	5	6	7	8	9	10	11
μ	0.5	0.55	0.6	0.65	0.67	0.7	0.75	0.8	0.85	0.9	1
Keim error	17.43	12.67	8.17	5.01	5.19	4.27	7.01	9.32	11.81	13.75	18.05

*In these tests we used 258 × 388 units, a neighborhood radius of 26, a learning rate of 0.1, 40 epochs and 20 000 input data points.

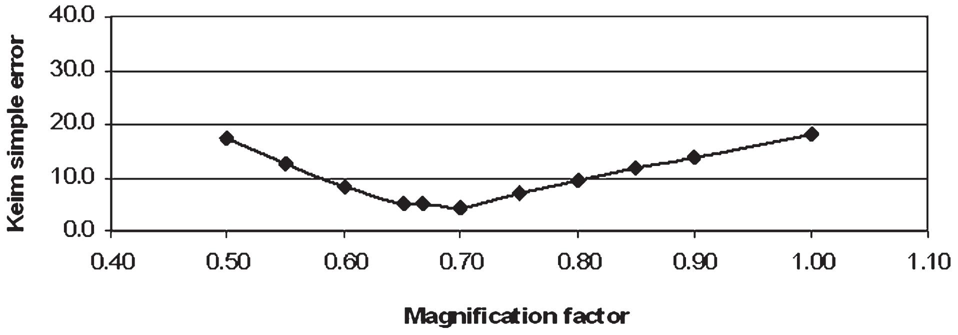


Figure 7. Carto-SOM error as a function of the magnification factor assumed for boosting original data.

predicted value of 2/3). From this result we conclude that boosting the original data using the magnification compensation of the input data points produces better results than using a proportional relation (μ=1).

5.2.3 SOM parameters evaluation. To test how the Carto-SOM methodology depends on the SOM parameters we performed a set of tests using different neighborhood radius r, learning rates α and number of epochs E.

5.2.3.1 Neighborhood radius. In Table 4 we present the 10 tests performed using different neighborhood radii and maintaining the remaining parameters.

As expected, if the value of neighborhood radius is very small then the SOM will have more difficulty adjusting to the input data, resulting in cartograms with higher errors (Figure 8). On the other hand, if we increase the neighborhood radius too much then the SOM will be less stable and the final results tend to have higher errors. In the tests performed we achieved an optimum error using a radius of 26 which corresponds to 10% of the number of units in a row.

5.2.3.2 Learning rate. We evaluate the effect of the learning rate by using 10 different values for this parameter while maintaining all the other parameters (Table 5).

Table 4. Variation of the neighborhood radius.*

Test	1	2	3	4	5	6	7	8	9	10
Radius	1	3	5	13	26	52	77	103	129	155
Keim error	36.61	30.76	22.72	4.87	4.55	6.58	6.95	7.97	9.68	11.22

*In these tests we used 258 × 388 units, a learning rate of 0.1, 40 epochs and 20 000 input data points.

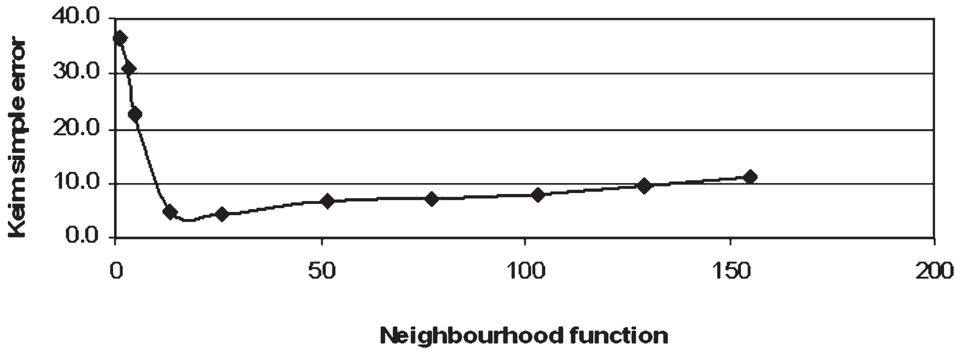


Figure 8. Carto-SOM error as a function of neighborhood radius.

Table 5. Variation on the learning rate.*

Test	1	2	3	4	5	6	7	8	9	10
Learning rate	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
Keim error	*4.55*	*5.69*	*4.5*	*4.87*	*5.5*	*6.59*	*4.36*	*4.11*	*4.86*	*4.23*

*In these tests we used 258 × 388 units, a neighborhood radius of 26, 40 epochs and 20 000 input data points.

In Figure 9 we present the Keim simple error for each cartogram using different learning rates. Although changing this parameter produces different cartogram errors these are not relevant and we conclude the learning rate is not a sensitive parameter.

5.2.3.3 Number of epochs. The third parameter tested was the number of epochs used in the SOM training phase. Again, 10 tests were executed to evaluate the influence of the number of epochs on the cartogram error (Table 6).

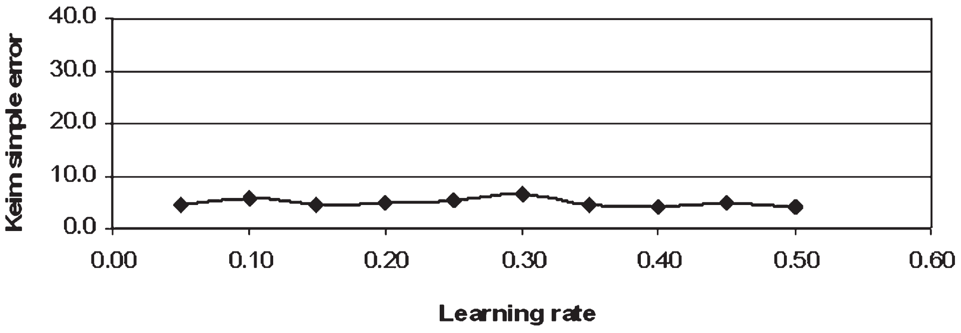


Figure 9. Carto-SOM error as a function of learning rate.

Table 6. Variation on the number of epochs.*

Test	1	2	3	4	5	6	7	8	9	10
Epochs	1	2	5	10	20	25	30	35	40	45
Keim error	*33.4*	*15.9*	*4.2*	*3.7*	*4.6*	*5.7*	*4.5*	*4.9*	*5.5*	*6.6*

*In these tests we used 258 × 388 units, a neighborhood radius of 26, a learning rate of 0.1 and 20 000 input data points.

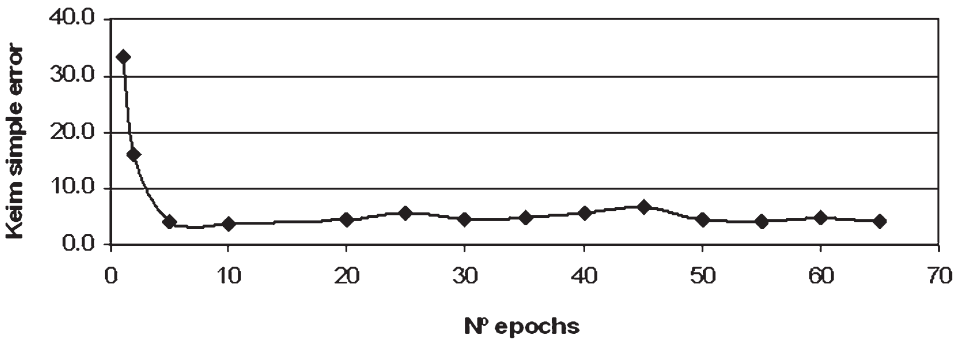


Figure 10. Carto-SOM error as a function of the number of epochs used.

In the following figure (Figure 10) we present the Keim simple error for the tests performed. As in the learning rate evaluation, the number of epochs used does not have a great influence in the final cartogram error, provided that the number of epochs is greater than five.

5.2.4 SOM dimension parameters. In these experiments we want to test how the Carto-SOM methodology depends on the SOM dimension parameters, *i.e.* how does the number of units and the number of input data points influence the error obtained in the cartograms.

5.2.4.1 Number of units. In this analysis we want to evaluate the cartogram errors when using different numbers of units (Table 7). The ratio between the number of rows and columns is the same as the ratio of the width and height of the original region of interest. In this case (which refers to the USA) a row/column ratio of 1/1.5 is used.

Figure 11 shows the errors obtained. As we can see there is a number of units (4988 units) for which the error tends to stabilize. There is reason to believe that the exact number of units for which the error stabilizes depends on the complexity of the regions being mapped.

5.2.4.2 Number of input data points. Finally we evaluate the effect of the number of input data points by performing ten tests using different numbers of input data points (Table 8).

Figure 12 presents the cartogram error for each test. We did not perform any tests with less than 8100 data points because we imposed that at least 10 data points should be generated for each region. If care is not taken to guarantee this, some regions will

Table 7. Variation on the number of units used.*

Test	1	2	3	4	5	6	7	8	9	10
Number of rows	18	26	37	58	82	115	183	258	408	577
Number of columns	28	38	54	86	122	174	273	388	613	867
Total number of units	504	988	1998	4988	10 004	20 010	49 959	100 104	250 104	500 259
Keim error	*27.16*	*22.82*	*13.22*	*4.65*	*6.04*	*5.18*	*4.61*	*4.75*	*4.27*	*4.97*

*In these tests we used a neighborhood radius of 26, a learning rate of 0.1, 40 epochs and 20 000 input data points.

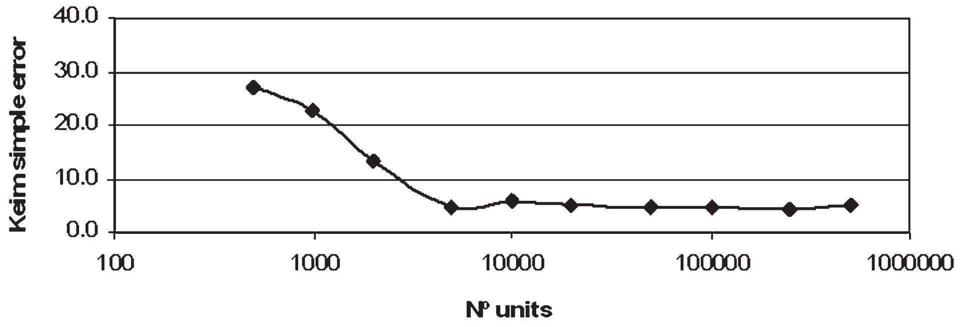


Figure 11. Carto-SOM error as a function of the number of units.

Table 8. Variation of the number of input data points.*

Test	1	2	3	4	5	6	7	8	9	10
nD	8100	9000	10 000	15 000	20 000	25 000	30 000	35 000	40 000	45 000
Keim error	*5.12*	*4.59*	*4.30*	*4.23*	*3.70*	*4.42*	*4.41*	*4.36*	*4.69*	*5.08*

*In these tests we used 258 × 388 units, a neighborhood radius of 26, a learning rate of 0.1 and 40 epochs.

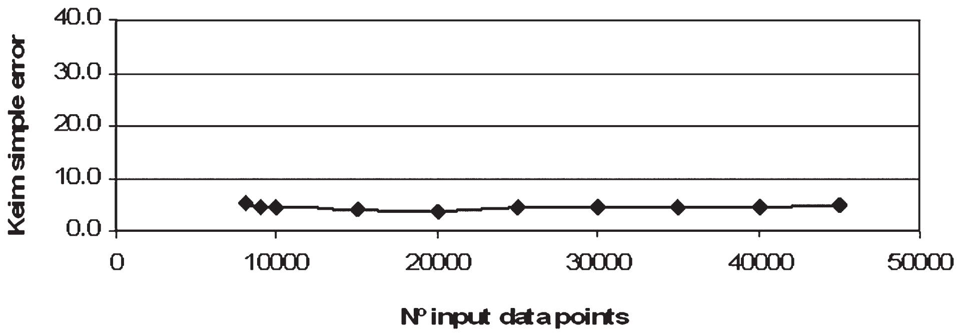


Figure 12. Carto-SOM error as a function of the number of input data points.

have a very high error because there simply are not enough data points to represent them. As long as this is taken into account, as we can see from this figure, the number of input data points has only a small influence on the cartogram error.

5.3 *Comparison between Carto-SOM, Dougenik and diffusion cartograms*

In this section we present the final cartograms for the three datasets; an artificial dataset, Portuguese 2001 population and USA 2000 population. We also compare the results of the Carto-SOM with the benchmark methods selected.

Figure 13 presents the artificial dataset and the resulting cartograms using the Carto-SOM, Dougenik and Gastner's methods. Visually these are good results since the cartograms represent the variable density by the size of each region while preserving topology, *i.e.* all areas have the same neighbors. Furthermore, preserving important points, such as the junction of all four regions in the center, increases the cartogram's recognition.

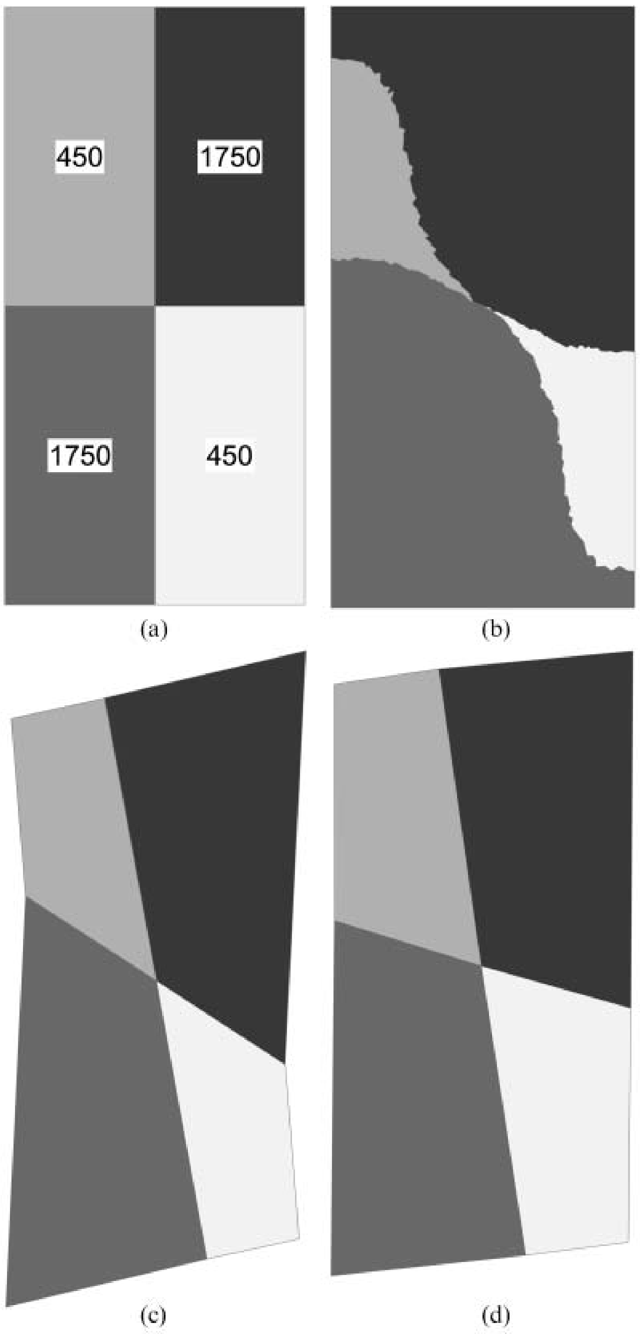
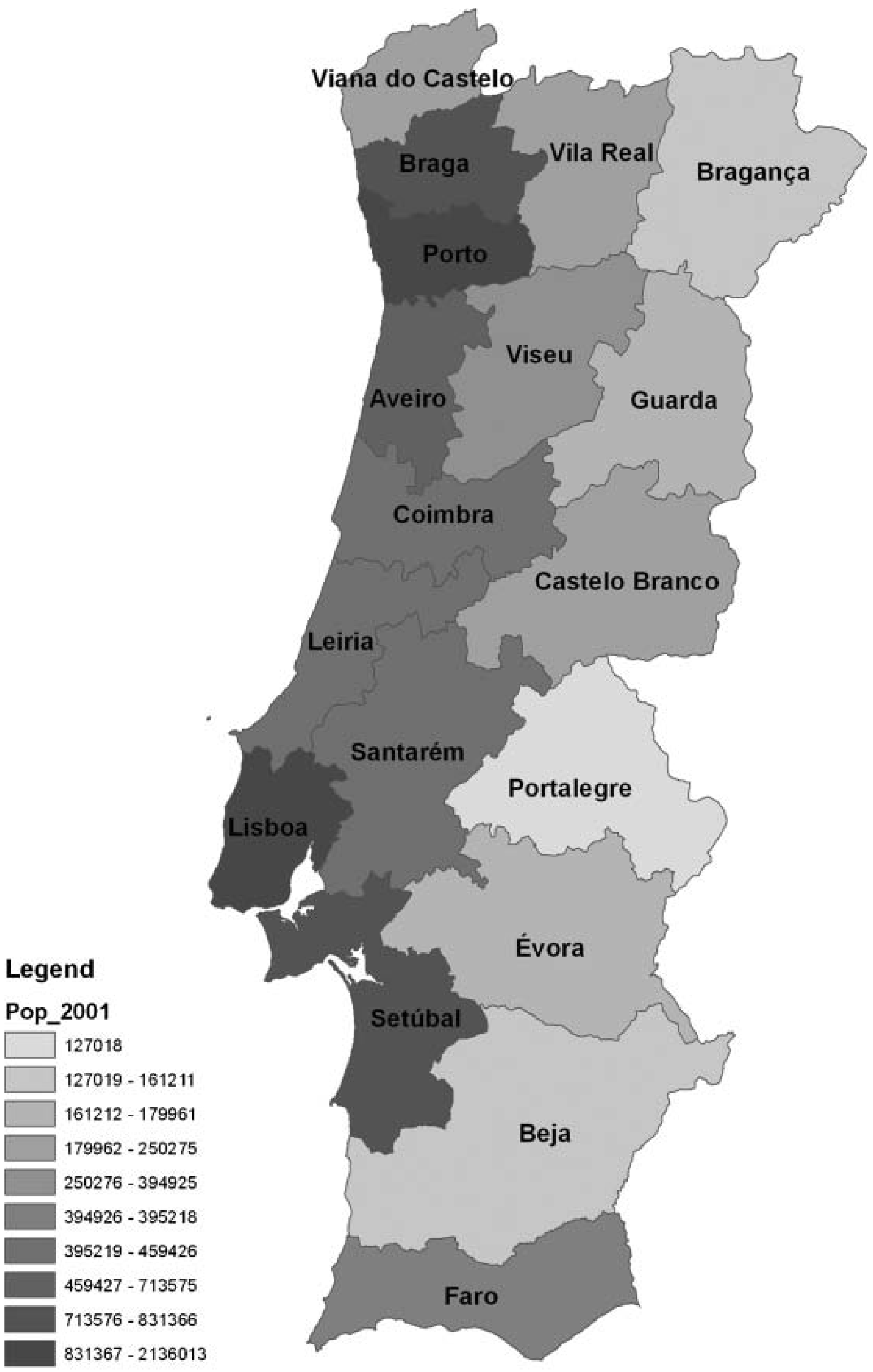


Figure 13. (a) Original map, (b) Carto-SOM, (c) Dougenik cartogram, and (d) diffusion cartograms of the artificial dataset.

Figure 14 presents Portuguese population cartograms using Carto-SOM, Dougenik and Gastner's methods.

In Figure 15 we present the USA 2000 population cartograms using the three different methodologies: Carto-SOM, Dougenik and Gastner's.



(a)

Figure 14. Portuguese population cartograms using the Carto-SOM, Dougenik and diffusion methods: (a) original 2001 Portuguese population map; (b) Carto-SOM; (c) Dougenik cartogram; (d) diffusion cartogram.

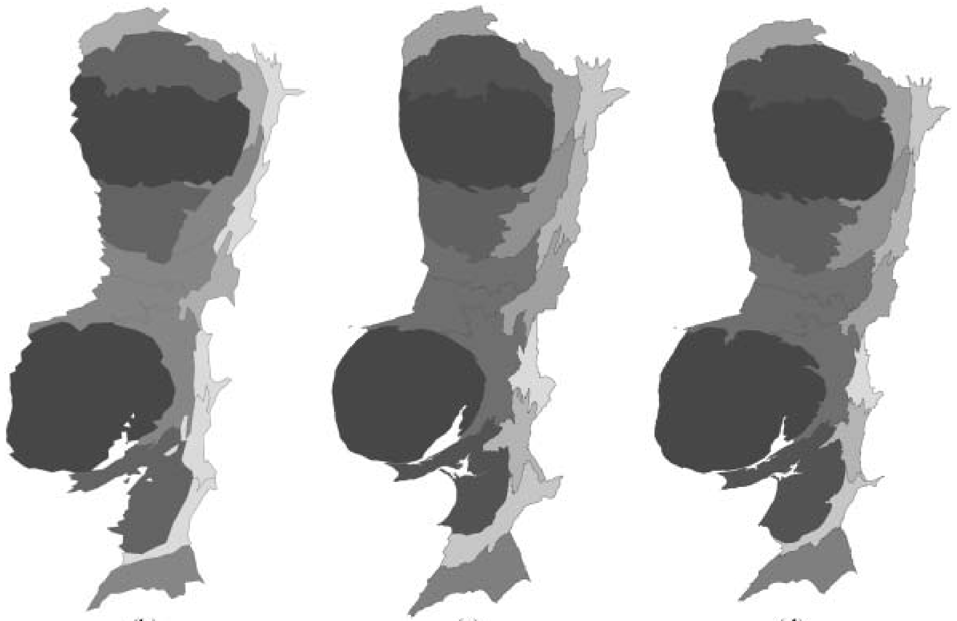


Figure 14. (*Continued.*)

Through visual analysis of the cartograms produced we can see that Carto-SOM manages to keep the shapes of individual regions in a way that is generally comparable to Dougenik and diffusion Cartograms. The Dougenik cartogram has the disadvantage of performing radial transformations on the shapes of the regions, making them circular as the number of iterations increases. This is clearly observed in Portugal, where Lisbon and Oporto appear almost as circles, and to a lesser extent in the USA where California tends to an egg shape. The diffusion cartogram also suffers from this bubble effect, albeit to a lesser extent. Because of the same effect in the Dougenik cartogram, some regions are unevenly squashed, making Texas look skewed (it looks straight in the Carto-SOM), while Setúbal and Santarém in Portugal are unevenly distorted by neighboring Lisbon. The Carto-SOM also manages to keep the overall shape of the cartogram similar to the geographical map, even though, as we shall see in the next section, the variable of interest is better represented. Unfortunately, there are some cases in which certain regions are 'broken up' into non-contiguous areas in the cartogram. To minimize this effect we may increase the number of neurons used, but no definite solution has been developed for this problem.

In Table 9 we present the cartogram errors calculated for each cartogram produced. The Carto-SOM performance measured by this error is significantly better except for the case of Portugal, where it is still quite competitive.

5.4 *Related issues*

An important factor in cartogram construction is the number of areas used. Several methods are able to produce good cartograms only if a small number of areas is used. One of the most famous methods that is capable of using a high number of objects is Gastner's diffusion cartograms. Using this method cartograms were built for

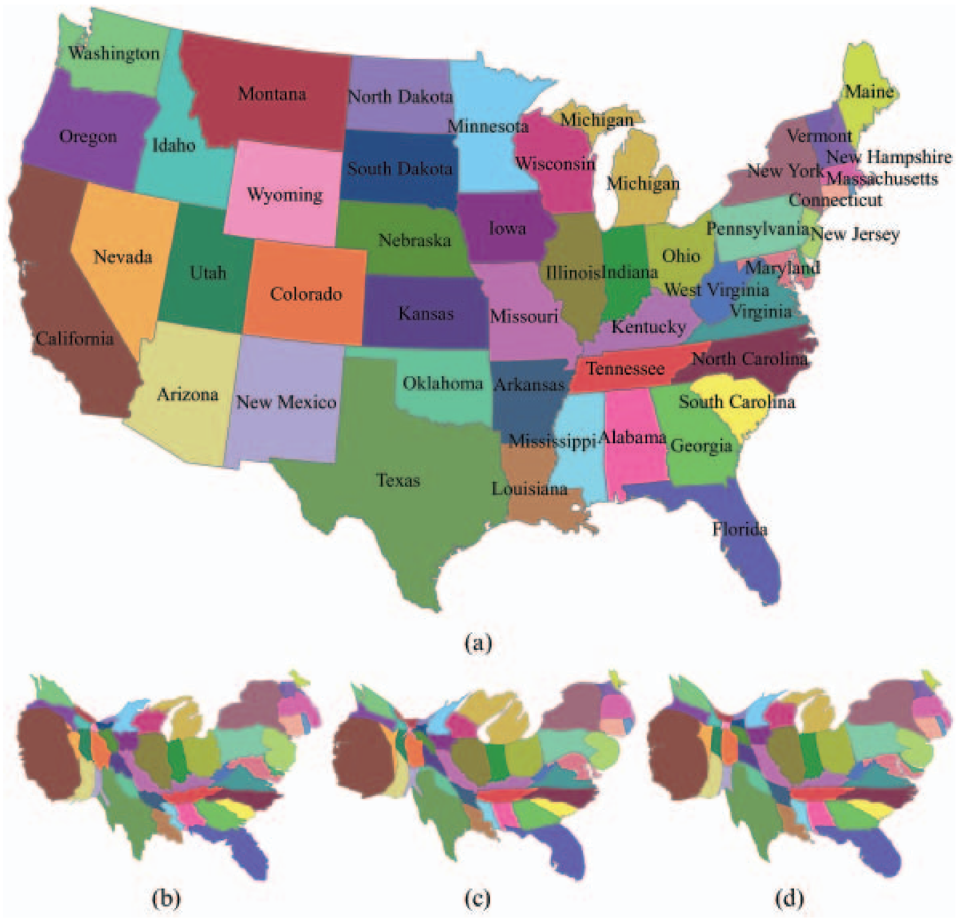


Figure 15. USA population cartograms using the Carto-SOM, Dougenik and diffusion methods: (a) original USA map; (b) Carto-SOM; (c) Dougenik cartogram; (d) diffusion cartogram.

Table 9. Keim error evaluation using different criteria on the various datasets.

	Artificial			USA			Portugal		
	se (%)	mqe (%)	we (%)	se (%)	mqe (%)	we (%)	se (%)	mqe (%)	we (%)
Dougenik	14.64	1.83	10.3	16.02	3.08	11.09	12.29	3.80	8.65
Diffusion	22.47	2.72	16.41	5.33	1.45	4.71	1.57	0.46	1.18
Carto-SOM	3.88	2.34	2.47	3.70	0.74	0.65	3.97	1.07	1.29

the population of the countries of the world (approx. 200 objects) and of the USA counties (approx. 3000 objects). The proposed Carto-SOM method can produce satisfactory cartograms for such large datasets, as is shown in Figures 16 and 17.

Another issue related with the use of a high number of areas to build a cartogram is the possibility of using parallel processing. The proposed Carto-SOM method can make very good use of parallel processing. One of the problems with parallel systems is

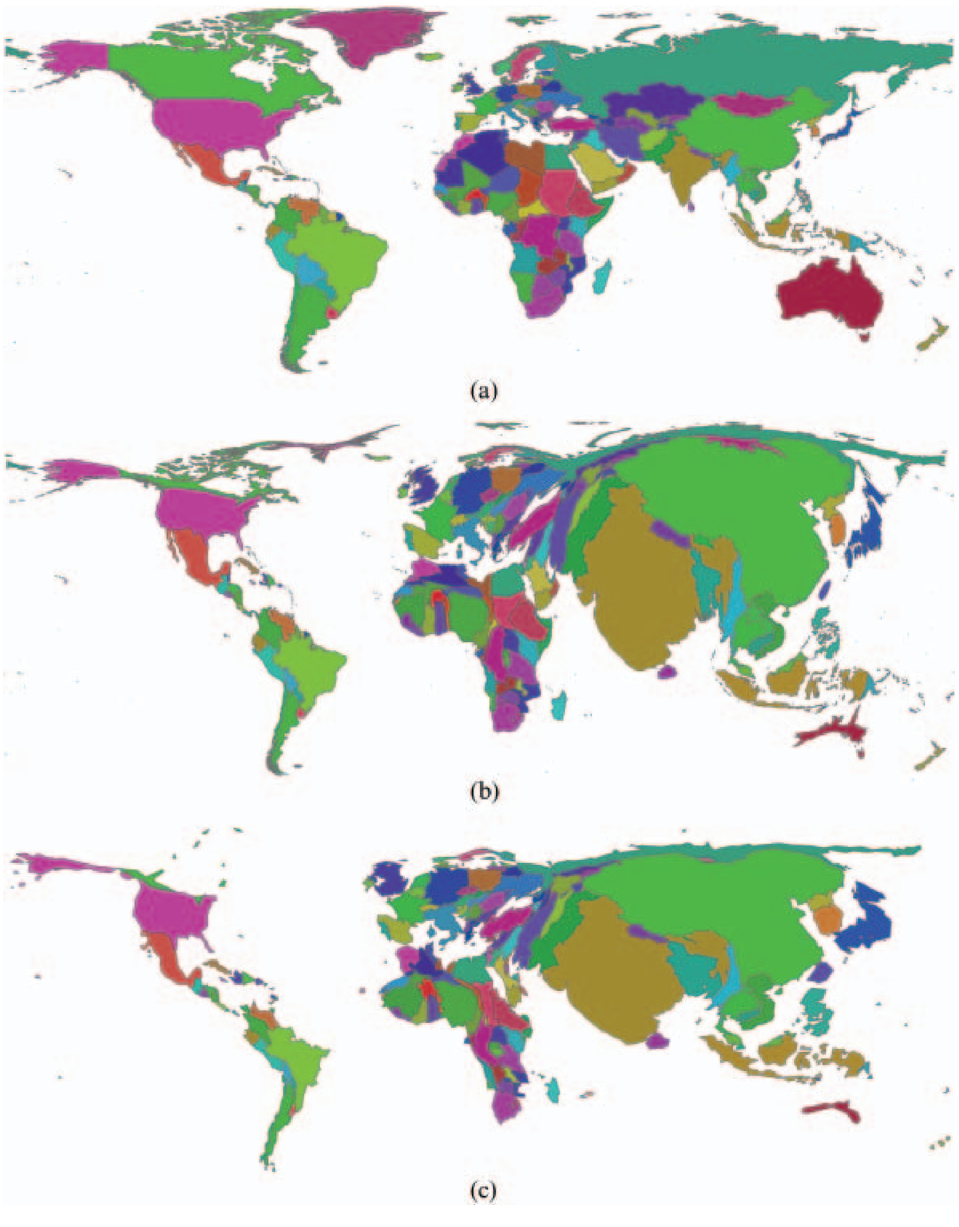


Figure 16. World countries population cartogram: (a) original world population map; (b) world population diffusion cartogram; (c) world population Carto-SOM cartogram.

that it is sometimes difficult to parallelize the algorithms, *i.e.* split the algorithm into different tasks requiring little interaction. Fortunately, the SOM training algorithm is easily adapted to parallel systems (Ultsch and Siemon 1990, Przytula *et al.* 1993). We may even use widely available networked PCs running Windows or Linux to efficiently train very large SOM's (Bandeira and Lobo 1998). This will not only allow us to drastically reduce the processing time but also to increase the number of SOM units that can be used, thus producing high quality cartograms in a short time.

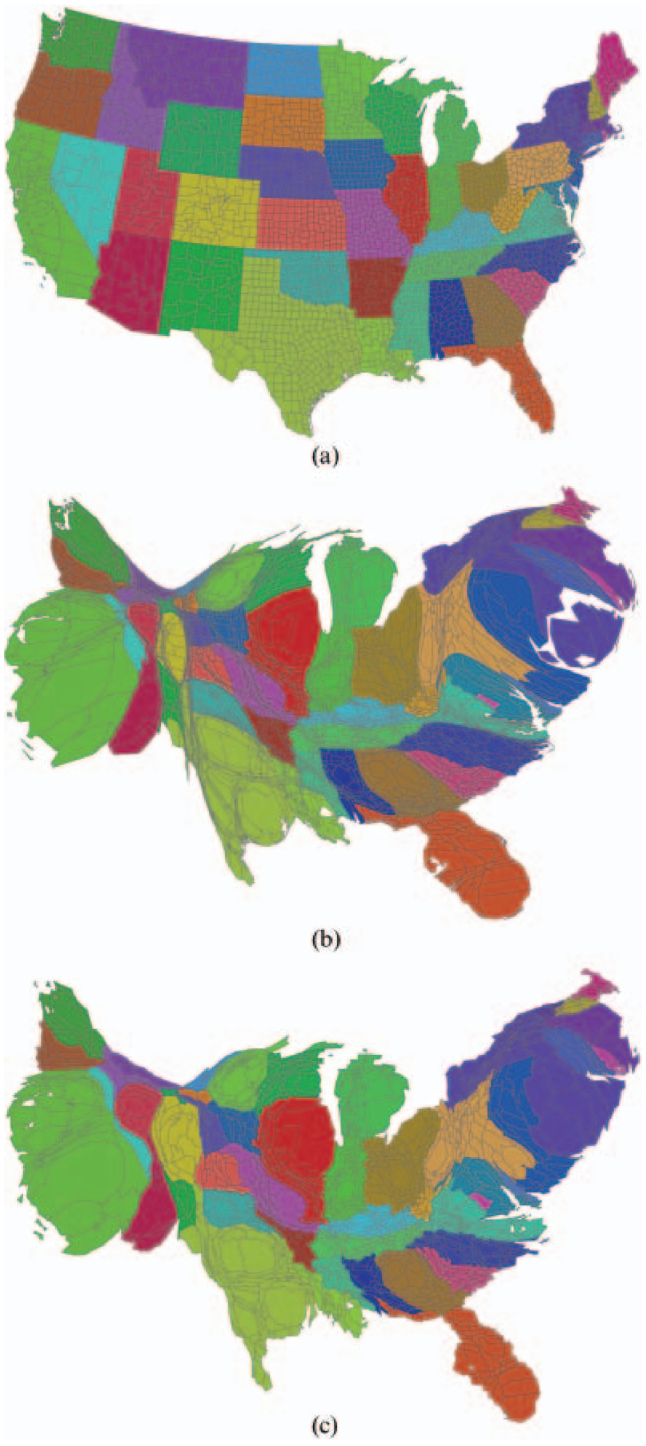


Figure 17. USA counties population cartogram: (a) original USA counties population; (b) USA counties diffusion cartogram; (c) USA counties Carto-SOM cartogram.

6. Conclusions

In this paper we presented a new method for building cartograms, called Carto-SOM, which is based on the SOM algorithm. To a certain extent this constitutes a completely different approach for building cartograms. In this case, instead of developing a new algorithm we make use of the widely available SOM algorithm and use it to build cartograms.

To evaluate the quality of the cartograms obtained with this method, tests were performed and assessed, comparing the proposed algorithm with existing ones, both using quantitative measures and visual inspection. The quantitative measure used was Keim's cartogram error. The tests suggest that the cartograms created using the Carto-SOM are good and accurate representations of the variables of interest. Visually we conclude that Carto-SOM is an efficient cartogram building algorithm, usually achieving better results than the Dougenik and Gastner's method used as benchmarks.

Finally, it must be emphasized that, using the Carto-SOM method, the only software necessary to create a cartogram is a standard implementation of the SOM algorithm. Such implementations are widely available, both in commercial data analysis programs and public domain packages.

7. Acknowledgments

The authors thank the anonymous referees for their valuable comments and constructive suggestions for improving this paper.

References

Bandeira, N. and Lobo, V., 1998, Training a self-organizing map distributed on a PVM network. In *IEEE World Conference on Computational Intelligence*, 4–9 May 1998, Anchorage, AL (New York: IEEE), pp. 35–39.

Bauer, H.-U., Der, R. and Herrmann, M., 1996, 'Controlling the magnification factor of self-organizing feature maps.' *Neural Computation*, **8**, pp. 757–771.

Bertin, J., 1983, *Semiology of graphics: diagrams, networks, maps* (Madison, WI: University of Wisconsin Press).

Du, C. and Liu, L., 1999, Constructing contiguous area cartogram using arcview avenue. In *The Proceedings of Geoinformatics'99 Conference*, 19–21 June 1999, Ann Arbor, MI (Michigan: The Association of Chinese Professionals in GIS), pp. 1–7.

Cottrell, M., Fort, J.C. and Pages, G., 1998, Theoretical aspects of the SOM algorithm. *Neurocomputing, Elsevier*, **21**, pp. 119–138.

Dersch, D.R. and Tavan, P., 1996, Asymptotic level density in topological feature maps. *IEEE Transactions on Neural Networks*, **6**(1), pp. 230–236.

Dougenik, J., Chrisman, N. and Niemeyer, D., 1985, An algorithm to construct continuous area cartograms. *Professional Geographer*, **37**, pp. 75–81.

Fort, J.C., 2006, SOM's mathematics. *Neural Networks*, **19**(6–7), pp. 812–816.

Fu, L., 1994, *Neural Networks in Computer Intelligence* (Singapore: McGraw Hill).

Gastner, M., 2005, 'Diffusion cartogram – cartogram code. Available online at: http://www.santafe.edu/~mgastner/ (accessed 02 May 2007).

Gastner, M. and Newman, M.E.J., 2004, Diffusion-based method for producing density-equalizing maps. *Proceedings of the National Academy of Sciences of the United States of America*, **101**(20), pp. 7499–7504.

Goldschmidt, R. and Passos, E., 2005, *Data Mining – Um guia prático* (Rio de Janeiro: Elsevier).

Gusein-Zade, S. and Tikunov, V., 1993, A new technique for constructing continuous cartograms. *Cartography and Geographic Information Systems*, **20**(3), pp. 167–173.

HEILMANN, R., KEIM, D.A., PANSE, C. and SIPS, M., 2004, RecMap: rectangular map approximations. *IEEE Symposium on Information Visualization*, 10–12 October 2004, Austin, TX (Austin, TX: IEEE), pp. 33–40.

HENRIQUES, R., 2006, Cartogram creation using self-organizing maps. *ISEGI* (Lisbon: New University of Lisbon), 144 p.

HOUSE, D. and KOCMOUD, C., 1998, Continuous cartogram construction. In *Proceedings of IEEE Visualization*, 18–23 October 1998, Research Triangle Park, NC (Research Triangle Park, NC: IEEE), pp. 197–204.

KASKI, S. and KOHONEN, T., 1998, Methods for interpreting a self-organized map in data analysis. In *ESANN '98, 6th European Symposium on Artificial Neural Neural Networks*, 22–24 April 1998, Brussels, Belgium (Brussels: D-Facto), pp. 185–190.

KASKI, S. and LAGUS, K., 1996, Comparing self-organizing maps. In ICANN96, 16–19 July 1996, Bochum, Germany (Berlin, Heidelberg, New York: Springer), pp. 809–814.

KEIM, D.A., NORTH, S.C. and PANSE, C., 2004, CartoDraw: a fast algorithm for generating contiguous cartograms. *IEEE Transactions on Visualization and Computer Graphics*, **10**(1), pp. 95–110.

KEIM, D.A., NORTH, S.C. and PANSE, C., 2005, Medial-axes based cartograms. *IEEE Computer Graphics and Applications*, **25**(3), pp. 60–68.

KEIM, D.A., NORTH, S.C., PANSE, C. and SCHNEIDEWIND, J.O., 2002, Efficient cartogram generation: a comparison. In *Proceedings of the IEEE Symposium on Information Visualization 2002*, 28–29 October 2002, Boston, MA (Washington, DC: IEEE Computer Society Press), 33 p.

KOCMOUD, C., 1997, *Constructing continuous cartograms: a constraint-based approach*. Maters thesis. Texas A&M University.

KOHONEN, T., 1982, Clustering, taxonomy, and topological maps of patterns. In *Proceedings of the 6th International Conference on Pattern Recognition*, 19–22 October 1982, Munich, Germany (Piscataway, NJ: IEEE Computer Society Press), pp. 114–125.

KOHONEN, T., 1995, *Self-organizing maps* (Berlin, Heidelberg: Springer).

KOHONEN, T., 2001, *Self-organizing maps* (Berlin, Heidelberg: Springer).

KOHONEN, T., HYNNINEN, J., KANGAS, J. and LAAKSONEN, J., 1995, *The self-organizing map program package* (Helsinki: Laboratory of Computer and Information Science, Helsinki University of Technology), 27 p.

MAENOU, T., FUJIMURA, K. and KISHIDA, S., 1997, Optimizations of TSP by SOM method. Progress in connectionsist-based information systems. In *Proceedings of the 1997 International Conference on Neural Information Processing and Intelligent Information Systems*, 26–30 November 1997, Dunedin, New Zealand (Singapore: Springer), Vol. 2, pp. 1013–1016.

MERENYI, E., JAIN, A. and VILLMANN, T., 2007, Explicit magnification control of self-organizing maps for 'forbidden' data. *IEEE Transactions on Neural Networks*, **8**(3), pp. 786–797.

OLSON, J., 1976, Noncontiguous area cartograms. *The Professional Geographer*, **28**(4), pp. 371–380.

PRZYTULA, K.W., PRASANNA, V.K., KUMAR, V.K.P. and PRAYTULA, K.W., 1993, *Parallel Digital Implementations of Neural Networks* (Upper Saddle River, NJ: Prentice-Hall).

RITTER, H., 1991, Asymptotic level density for a class of vector quantization processes. *IEEE Transactions on Neural Networks*, **2**(1), pp. 173–175.

RITTER, H. and SCHULTEN, K., 1988, Extending Kohonens self-organizing mapping Algorithm to learn ballistic movements. In *Neural computers*, R. Eckmiller and C. von der Malsburg (Eds) (Heidelberg: Springer-Verlag), pp. 393–406.

SARAJEDINI, B.A. and CHAU, P.M., 1996, Cumulative distribution estimation with neural networks. In *WCNN'96*, 15–20 September 1996, San Diego, CA (Mahwah, NJ: Laurence Erlbaum), pp. 876–880.

Skupin, A., 2003, A novel map projection using an artificial neural network. In *21st International Cartographic Conference*, 10–16 August 2003, Durban, South Africa (Durban: International Cartographic Association), pp. 1165–1172.

Takatsuka, M., 2001, An application of the self-organizing map and interactive 3-D visualization to geospacial data. In *GeoComputation '01 (6th International Conference on GeoComputation)*, 24–26 September 2001, Brisbane, Australia (Brisbane: University of Queensland).

Tobler, W., 1970, A computer model simulating urban growth in the Detroit region. *Economic Geography*, **46**, pp. 234–240.

Tobler, W., 1973, A continuous transformation useful for districting. *Annals, New York Academy of Sciences*, **219**, pp. 215–220.

Tobler, W., 1986, Pseudo-cartograms. *The American Cartographer*, **13**(1), pp. 43–50.

Tobler, W., 2004, Thirty-five years of computer cartograms. *Annals, Assoc. American Geographers*, **94**(1), pp. 58–73.

Ultsch, A. and Siemon, H.P., 1990, Kohonen's self-organizing neural networks for exploratory data analysis. In *Intl. Neural Network Conf. INNC90*, 9–13 July 1990, Paris, France (Paris: Kluwer Academic), pp. 305–308.

Vesanto, J., Himberg, J., Alhoniemi, E. and Parhankangas, J., 2000, *SOM Toolbox for Matlab 5* (Espoo: Helsinki University of Technology), 59 p.

Villmann, T., Merenyi, E. and Hammer, B., 2003, Neural maps in remote sensing image analysis. *Neural Networks*, **16**(3–4), pp. 389–403.