# Weighted Maps: Location-Aware Treemaps

Mohammad Ghoniem* Maël Cornil† Mickaël Stefas‡ Benoît Otjacques§

CRP – Gabriel Lippmann
41 rue du Brill, L-4422 Belvaux, Luxembourg

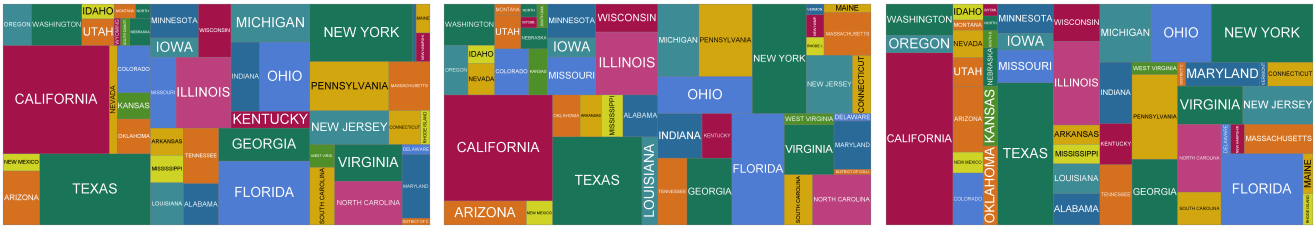


Figure 1: The USA population in 49 states according to the HistoMaps layout (left), the Weighted Maps layout (middle) and the Spatially Ordered Treemaps layout (right). All three algorithms generate positional anomalies.

## Abstract

The wide availability of quantitative geolocated structured data, e.g. census data, makes it desirable for classic tree visualizations to be location-aware. In this paper, we introduce a new treemap algorithm that produces rectangular cartograms consistently with the known location of data points on a geographic map.

**Index Terms:** H.5.2 [User Interfaces]: Graphical user interfaces (GUI)—Screen design;

## 1 Introduction

Broadly speaking, two main approaches are used to visualize hierarchical data: node-link diagrams and recursive enclosure of shapes (see the Treevis.net project [5] for a comprehensive list of techniques). This work falls in the second category where Treemaps [6] are the most well-known technique. Our purpose is to contribute to the research work that considers spatial properties of nodes in treemap layouts. To illustrate the problem, we consider the recursive split of a territory into administrative subdivisions. Such hierarchies can easily be represented by a treemap where the size of rectangles may encode a variety of abstract region attributes such as population. However, in the case of standard treemap layout algorithms, node position is not correlated to their known geolocations, undermining the ability of culturally trained map users to get a meaningful overview of the whole territory or to rapidly locate specific regions by using their prior knowledge of geography.

## 2 The Weighted Maps Algorithm

The Weighted Maps layout (WM) is based on the idea of slicing the display space into chunks (denoted $C_i$ in Figure) in which the data points are allocated sequentially from East to West, or from North to South, while maintaining weight/area proportionality. The slicing direction is chosen in order to yield chunks with the best aspect ratio (closer to 1). As data points are gradually allocated to a chunk and the sum of enclosed weights gets close to the chunks capacity, a decision has to be made as to expanding the chunk slightly to accommodate the last point or shrink it to match the weight of previously allocated points. Once the chunk's free edge (the western or the southern edge) is adjusted, all unallocated points are laid out in the next chunk. This splitting strategy is repeated recursively until each chunk holds a single data point. Below we walk through the algorithm on an illustrative example depicted in Figure 2.

Step 1: We start with a table $T$ containing seven geolocated data items $t_i$, that have additional attributes. First we determine the size $s_i$ of each item, based on the data attributes, e.g. population. The area $a_i$ for item $t_i$ in the layout will be proportional to its size $s_i$.

Step 2.a: The data items are laid out in an available area $A_k$, where $k$ represents the depth of recursion. The order in which the data items are laid out is based on the aspect ratio of $A_k$. In the example, $A_0.width > A_0.height$, thus the items in $T$ are sorted according to their x coordinate (or latitude).

Step 2.b: The algorithm determines the preferred area $C_{pref}$ of a chunk in $A_k$ based on $A_k.width$ and $A_k.height$. Instead of adding items to a chunk until the chunk aspect ratio reaches one (as in the squarified treemap approach), our approach chooses an aspect ratio for the chunk upfront which will be close to one but not necessarily exactly one. This prevents having, for example, two square chunks and one elongated chunk at the end. The algorithm tries to minimize the difference between the preferred and the actual area of a chunk. Hence, the first chunk $C_1$ is shrunk making its capacity $a_1$. $C_1$ contains just one item, therefore recursion ends here and $C_1$ can be laid out in $A_0$. The direction in which items are laid out (East to West or North to South) in each chunk in $A_0$ is based on the aspect ratio of $A_0$. Next, the algorithm proceeds in the same way for $C_2$ in step 3. In this case it recurses, though without leading to visual changes of the layout (step 5).

Step 4: Since the number of chunks is determined upfront, the remaining items are put in $C_3$.

Step 6: Here the algorithm recurses into $C_3$. Note that in step 6.3 $A_7$'s height is greater than its width, and therefore the layout direction becomes from North to South. When it returns from the recursion, the final WM is laid out as shown in step 7.

It is worth noting that at the beginning, depending on the aspect ratio of the root node, the available space may be partitioned in more than two equal sub-rectangles. In Figure 2, $A_0$ is split in three. However, in further recursion steps, the WM algorithm will eventually turn into a binary space partitioning algorithm.

*e-mail:ghoniem@lippmann.lu
†e-mail:cornil@lippmann.lu
‡e-mail:stefas@lippmann.lu
§e-mail:otjacque@lippmann.lu

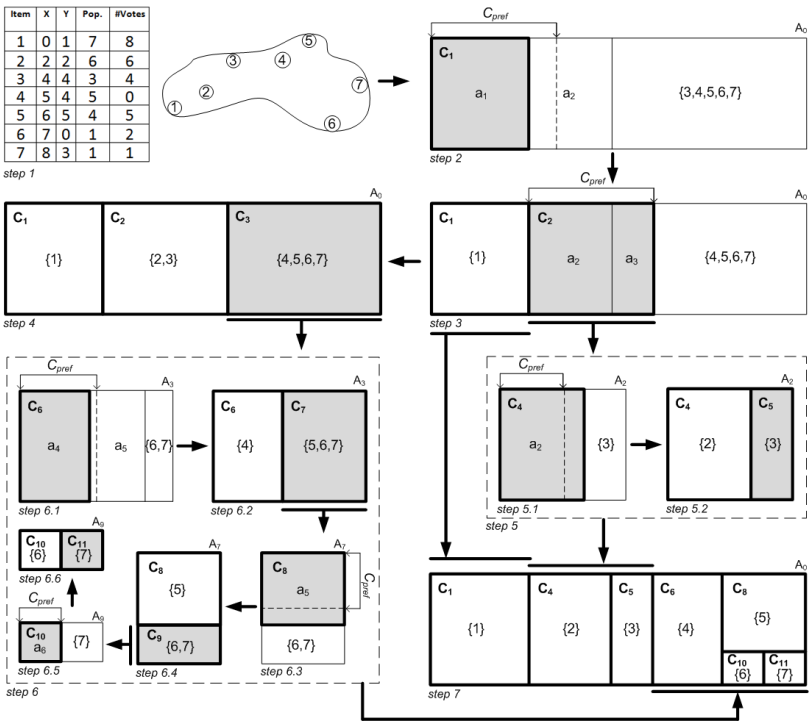


Figure 2: An illustration of the Weighted Maps layout on a flat hierarchy with 7 geolocated nodes.

In the case of multi-level hierarchies, the algorithms may be applied level by level to lay out top-level nodes first according to the algorithm explained previously, then recursively by laying out children nodes further down in their respective parent space.

## 3 Related Work

The need for location-aware tree representations has previously been addressed in the information visualization community using treemap variants. The Histomap algorithm (HM) was used to visualize email server logs taking into account the geographic origin of emails [4]. HM uses a binary partitioning scheme of the data points set which splits the latitude/longitude range in halves recursively in order to determine the position of rectangles. The Spatially Ordered Treemaps algorithm (SOT) addresses the same problem [9]. Building on the *squarified treemaps* algorithm [1], instead of simply taking the nodes in turn based on data order, they place the nodes based on their distance to the enclosing rectangle to be filled. A computational study compared SOT to HM with respect to their average aspect ratio, average distance displacement and average angular displacement. Beyond flat cartogram generation, both SOT and HM can handle multi-level trees [7].

The inclusion of spatial constraints into treemaps puts our work in the substantial amount of research about rectangular cartograms. Tobler's extensive review of cartograms [8] summarizes the work in this domain. It points out the preservation of neighborhood relationships as a useful quality criterion. Focusing on adjacency preservation, Buchin et al. [2] use a graph structure to formulate the cartogram generation problem as a graph optimization problem. Priority is given to adjacency preservation, by compromising on the proportionality between node weights and rectangle areas. Other recent work explore the use of grid layouts of geographic subdivisions. For instance, Eppstein et al. [3] model the problem as a point set matching optimization problem. An extensive review of cartogram generation is however beyond the scope of this paper.

WM differs from HM insofar that the latter uses the middle of the longitude/latitude range as a pivot. In the WM layout, no compromise is made on node weights, and aspect ratio is explicitly optimized. Geolocation and relative orientation are of prime importance in both. It differs from multi-objective optimization-based cartogram generation approaches with respect to the high computational cost that such techniques incur, at the expense of interactivity.

## 4 Perspectives

Early tests reveal significant differences between the three location-aware treemap layouts regarding large flat cartograms. Future work includes a thorough comparative study of these algorithms based on real and simulated datasets to assess their scalability, as well as user experiments for usability assessment.

## References


[1] M. Bruls, K. Huizing, and J. J. Wijk. Squarified treemaps. In *Data Visualization 2000*, Eurographics, pages 33–42. Springer Vienna, 2000.

[2] K. Buchin, D. Eppstein, M. Löffler, M. Nöllenburg, and R. I. Silveira. Adjacency-preserving spatial treemaps. In *Algorithms and Data Structures*, volume 6844 of *LNCS*, pages 159–170. Springer, 2011.

[3] D. Eppstein, M. van Kreveld, B. Speckmann, and F. Staals. Improved grid map layout by point set matching. In *PacificVis 2013*, pages 25–32.

[4] F. Mansmann, D. Keim, S. North, B. Rexroad, and D. Sheleheda. Visual analysis of network traffic for resource planning, interactive monitoring, and interpretation of security threats. *IEEE TVCG*, 13(6):1105–1112, 2007.

[5] H.-J. Schulz. Treevis.net: A tree visualization reference. *IEEE Computer Graphics and Applications*, 31(6):11–15, 2011.

[6] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99, Jan. 1992.

[7] A. Slingsby, J. Dykes, and J. Wood. Rectangular hierarchical cartograms for socio-economic data. *J. of Maps*, 6(1):330–345, 2010.

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

[9] J. Wood and J. Dykes. Spatially ordered treemaps. *IEEE TVCG*, 14(6):1348–1355, 2008.