Necklace Maps

Bettina Speckmann and Kevin Verbeek

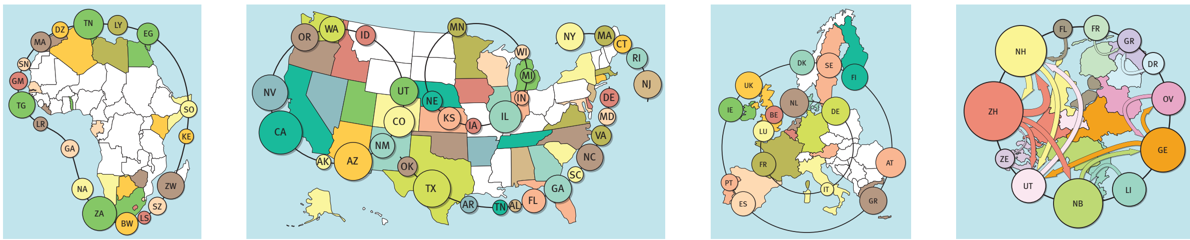

Fig. 1. Necklace maps: Internet users in Africa, illegal immigrants, gender pay gap, and population together with relocation.

Abstract—Statistical data associated with geographic regions is nowadays globally available in large amounts and hence automated methods to visually display these data are in high demand. There are several well-established thematic map types for quantitative data on the ratio-scale associated with regions: choropleth maps, cartograms, and proportional symbol maps. However, all these maps suffer from limitations, especially if large data values are associated with small regions. To overcome these limitations, we propose a novel type of quantitative thematic map, the necklace map. In a necklace map, the regions of the underlying two-dimensional map are projected onto intervals on a one-dimensional curve (the necklace) that surrounds the map regions. Symbols are scaled such that their area corresponds to the data of their region and placed without overlap inside the corresponding interval on the necklace. Necklace maps appear clear and uncluttered and allow for comparatively large symbol sizes. They visualize data sets well which are not proportional to region sizes. The linear ordering of the symbols along the necklace facilitates an easy comparison of symbol sizes. One map can contain several nested or disjoint necklaces to visualize clustered data. The advantages of necklace maps come at a price: the association between a symbol and its region is weaker than with other types of maps. Interactivity can help to strengthen this association if necessary. We present an automated approach to generate necklace maps which allows the user to interactively control the final symbol placement. We validate our approach with experiments using various data sets and maps.

Index Terms—Geographic Visualization, Automated Cartography, Proportional Symbol Maps, Necklace Maps.

1 Introduction

A significant part of cartographic map production is dedicated to *thematic maps*, that is, maps that visualize attributes or concepts (as opposed to topography). Statistical data associated with collections of geographic regions (like countries, states, or provinces) is nowadays globally available in large amounts and hence automated methods to visually display this kind of data are in high demand. Typical data sets are population, income, or production of various goods; a good (quantitative) thematic map strives to visualize the spatial variation of the data from place to place.

There are several well-established thematic map types for quantitative data on the ratio-scale associated with regions, specifically, *choropleth maps*, *cartograms*, and *proportional symbol maps*. However, all these maps suffer from limitations. Choropleth maps tend to overemphasize large regions and can generally only be used for data that is uniformly distributed within each region. Cartograms deform the underlying regions according to the data, which can make the map virtually unrecognizable if the data value differs greatly from the original area of a region or if data is not available at all for a particular region. Finally, proportional symbol maps can appear very cluttered with many overlapping symbols if large data values are associated with small regions. In such cases, the underlying map is hardly visible, making it difficult to associate symbols with the correct regions.

In this paper we propose a novel type of quantitative thematic map, called *necklace map*, which overcomes the limitations described above. Necklace maps combine elements of proportional symbol maps and boundary labeling. The underlying two-dimensional map is projected onto a one-dimensional curve (the necklace) that surrounds the map regions. The projection maps each region of the input map to a contiguous interval on the necklace in such a way, that the interval captures the global location of the region with respect to the necklace. Just as with proportional symbol maps, a symbol (most commonly a disk or a square) is scaled such that its area corresponds to the data for a particular region. The symbol is then placed inside the corresponding interval on the necklace. We show how to optimize the symbol sizes such that all symbols are as large as possible and can be placed without overlap inside their intervals.

Necklace maps appear clear and uncluttered and allow for comparatively large symbol sizes. They visualize data sets well which are not proportional to region sizes and which do not contain data for all regions of the input map. The linear ordering of the symbols along the necklace facilitates an easy comparison of symbol sizes. Necklace maps also allow for a simple integration of flows between symbols and are suitable to display multi-variate data. Finally, one map can contain several nested or disjoint necklaces to visualize clustered data. The advantages of necklace maps come at a price: the association between a symbol and its region is weaker than with other types of maps. Interactivity can help to strengthen this association if necessary.

We present an automated approach to generate necklace maps. Optimizing the symbol sizes is an NP-hard problem for which we implemented an exact algorithm that is fixed-parameter tractable in the *thickness* of the region intervals on the necklace. That is, our algorithm scales to an arbitrary number of symbols per necklace, as long as no point of the necklace is covered by more than 15 region intervals. In practice this means that a map of the U.S. states can be computed in a few seconds, and a world map is feasible if continents are assigned different necklaces. There are several optimization choices for the final placement of the symbols which can be controlled interactively by

- *Bettina Speckmann is with TU Eindhoven, E-mail: speckman@win.tue.nl.*
- *Kevin Verbeek is with TU Eindhoven, E-mail: k.a.b.verbeek@tue.nl.*

Manuscript received 31 March 2010; accepted 1 August 2010; posted online 24 October 2010; mailed on 16 October 2010.
For information on obtaining reprints of this article, please send email to: tvcg@computer.org.

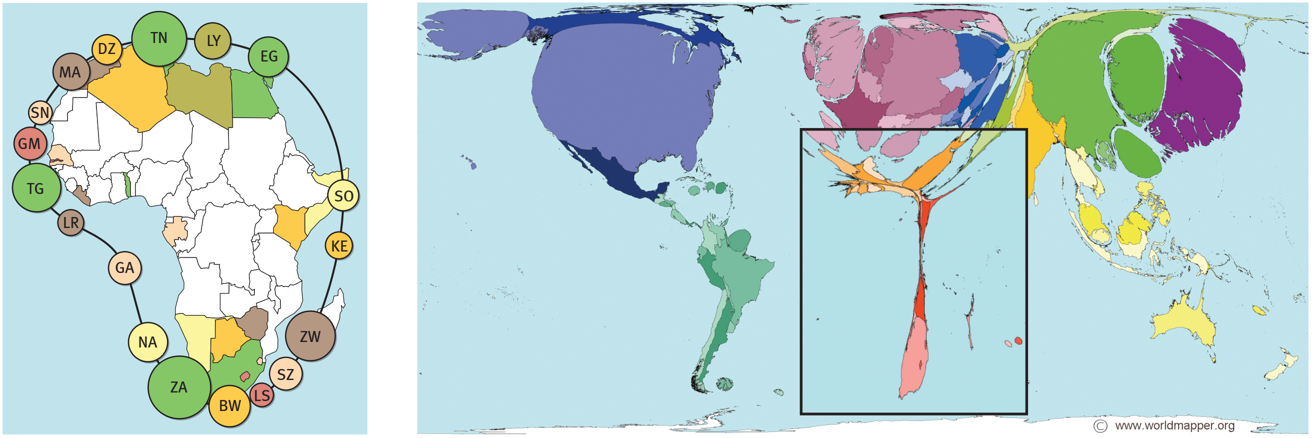

Fig. 2. Internet users in Africa in 2002 per thousand inhabitants. Left: a necklace map, displaying only regions with a value of at least ten. Right: a world cartogram, Africa is magnified. © Copyright 2006 SASI Group (University of Sheffield) and Mark Newman (University of Michigan).

the user. We showcase the results of experiments with various data sets and maps using one or more nested or disjoint necklaces.

The remainder of the paper is organized as follows. Section 2 discusses related work and visually compares necklace maps to proportional symbol maps and cartograms. Section 3 formally defines necklace maps and their quality criteria. Section 4 presents our algorithm to compute necklace maps with a single necklace. Section 5 discusses various extensions: necklace maps with multiple necklaces, either nested or disjoint, integration of flows between symbols or between necklaces, and options for the display of multi-variate data. Section 6 gives some technical details about our implementation and we close in Section 7 with a discussion of our work.

2 Related work

Choropleth maps are probably the most commonly used type of thematic map. The regions of the map are shaded or patterned according to their data values. Choropleth maps are a very intuitive mapping technique, the association between regions and data values is immediate. However, choropleth maps are ill-suited to visualize absolute values: users tend to mentally integrate over the region areas and will interpret all data as densities. Furthermore, large regions tend to be overemphasized. Choropleth maps generally should be used only for regions of near-uniform size and shape and for data that is uniformly distributed within each region. See the books by Dent [3] and Slocum et al. [13] for a detailed discussion of these issues.

Cartograms or *value-by-area maps* scale the regions of the input map such that the area of each region represents its data value. There are several different types of cartograms. The standard type—the contiguous area cartogram—has deformed regions so that the desired sizes can be obtained and the adjacencies kept. Various algorithms have been proposed to create this kind of cartogram [5, 6, 7, 9, 10, 16]. Contiguous area cartograms perform best if the data values are somewhat related to the area of the input regions. Since each region must be shown, it is difficult to incorporate missing data or use thresholding to concentrate on a meaningful subset. If the data values are significantly smaller or larger than the original region sizes, then the map might become virtually unrecognizable. See, for example, Fig. 2, which shows Internet users in Africa. The cartogram (which shows this data for the whole world) makes it very clear that there are few internet users in Africa, however, the (thresholded) necklace map lets us see where the few users actually are. (Note the comparatively small country Gambia which has a significant number of internet users.) Also, it is generally hard to judge the size of regions in contiguous area cartograms.

Circular cartograms, introduced by Dorling [4], represent each region with a circle which is scaled such that its area corresponds to the data value. The circles are placed without overlap and in such a way that region adjacencies and relative positions are maintained as well as possible. Such a placement is often achieved via iterative methods and the results are not always satisfactory. Symbols might be moved far from their initial position and in arbitrary directions, making it difficult to find the symbol of a particular region. See, for example, Fig. 15 bottom, which shows the Olympic medal count. The circle of Italy lies above Switzerland and France is in the south-east of the Netherlands.

Rectangular cartograms, introduced by Raisz [12], represent each region by a rectangle. Rectangles and circles both have the advantage that the sizes (area) of the regions can be estimated comparatively well. However, the rectangular shape is not very recognizable and it imposes limitations on the possible layout, making it difficult to maintain region adjacencies. There are several algorithms to construct rectangular cartograms [11, 14]. Other related approaches which loosen the adjacency requirements but keep some spatial proximity of the rectangles are *rectangular map approximations* [8] and *spatial treemaps* [18].

Proportional symbol maps or *graduated symbol maps* place scaled symbols or diagrams directly on the input map, often on the centroid of the regions. The symbol, most commonly a disk or a square, is scaled such that its area corresponds to the data value of the region. Alternatively, diagrams (like pie charts) can be used instead of symbols to show additional data. Algorithms for symbol maps and diagram placement are given in [2, 17]. Since symbols tend to have simple shapes, their areas can be estimated comparatively easily. However, overlapping symbols or large symbols associated with small regions can make it difficult to determine which region a symbol is associated with and to accurately judge its size. See, for example, Fig. 3, which shows the GDP drop in parts of Western Europe. Symbol sizes for both the necklace and the symbol map are equal, but it is significantly harder to compare symbols sizes accurately with the symbol map. In an in-

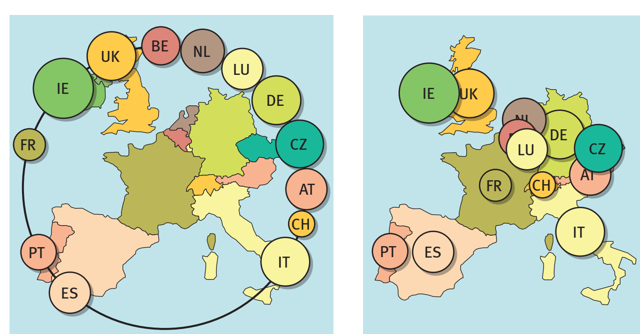

Fig. 3. GDP drop in 2009 as percentage change on 2008. Left: a necklace map. Right: a proportional symbol map, symbols are placed on region centroids.

teractive setting, morphing from one map to the other might give the user both clarity and increased spatial association.

Boundary labeling is frequently used in medical and technical drawings. Comparatively large labels are placed around an axis-parallel rectangle that contains the points or areas to be labeled. Each label is connected to its site with a polygonal line, a *leader*, and no two leaders intersect. Labels tend to have the same (fixed) height and the main algorithmic problem is to find the optimal order of labels around the rectangle. This question was first studied by Bekos et al. [1]. Necklace maps contain some elements of boundary labeling, insofar, that the symbols are placed close to the boundary of the input map. However, symbol sizes are neither equal, nor fixed, and geographic constraints for symbol placements have to be obeyed. Furthermore, necklace maps do not need leaders, since the association between region and symbol is created via spatial proximity and color coding.

3 Necklace maps

A necklace map is a thematic map and as such consists of two components: an underlying geographic map and the thematic overlay. Necklace maps are intended to display quantitative data on the ratio-scale associated with regions. (Data on the ratio scale has a meaningful zero [13].) The thematic overlay consists of symbols which are associated with the regions of the input map. Just as with symbol maps each symbol–most commonly a disk or a square, although other shapes are possible–is scaled such that its area corresponds to the data value associated with its region. The symbols are then placed without overlap on one or more necklaces: curves that enclose the regions of the input map. A good necklace map preserves as much of the spatial relation between a symbol and its region as possible. Each (two-dimensional) region of the input map is projected to a contiguous (one-dimensional) interval on a necklace in such a way that the global location of the region with respect to the necklace is preserved (see Fig. 4 left). Symbols are restricted to lie with their centers within their interval on the necklace. See Fig. 4 right: the center of the necklace lies in France and the symbols are encountered in approximately the same order and direction as their regions lie with respect to the necklace center.

The shape of the necklaces is an important factor in the visual appearance of a necklace map. Circles and ellipses are highly symmetric geometric shapes that detract little attention from the symbol sizes. They leave the interior of the map largely unoccupied and hence the geographic information of the base map visible. Furthermore, such necklace shapes allow for an easy integration with choropleth maps or flows. However, it can also be desirable to use necklace shapes that are similar to the global shape of the input regions (see, for example, Fig. 2 left) or which are not closed. In principle any *star-shaped* open or closed curve can be used as a necklace, however, our current implementation is restricted to cubic B-splines. A curve is star-shaped if there exists at least one point that can see the complete curve—this point, the center of the necklace which necessarily lies in the interior in case of a closed curve, is needed to define the projection from the input regions onto the necklace.

A necklace map communicates its message via the sizes of its

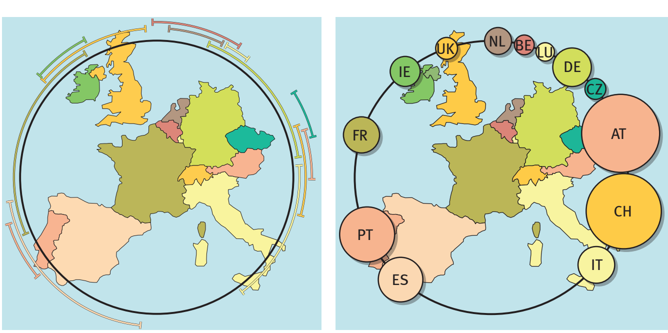

Fig. 4. Left: region intervals on a necklace. Right: energy consumption from renewable resources in 2007.

symbols—both their actual size and the ratio between sizes. A significant body of work (theoretical as well as user studies) discusses which sizing communicates the difference between quantities most effectively, see the books by Dent [3] and Slocum et al. [13] for details. Three basic types of scaling are the following: *mathematical scaling* sizes the areas of the symbols in direct relation to the data, *perceptual scaling* enlarges larger symbols beyond their mathematically correct size to compensate for humans inability to judge the relative sizes of area symbols accurately, and *range grading* subdivides the data into classes. All necklace maps in this paper use mathematical scaling.

Quality criteria. The quality of a necklace map is determined by several factors. Recall that the center of each symbol is restricted to lie inside its interval on the necklace.

- Good symbol positions—the intervals on the necklace capture the spatial relation of regions and necklace well.
- Maximal symbol size.
- Disjoint symbols.
- Suitable order of symbols along necklace—symbols which are neighbors on the necklaces correspond to neighboring regions.

There is a clear trade-off between the first two quality criteria: large intervals enable large symbol sizes in exchange for inferior symbol positions. In one extreme, each interval is the whole necklace, so symbol placement is arbitrary and symbol size can easily be maximized. However, most spatial associations between symbols and areas will be lost. In the other extreme each symbol location on the necklace is specified as close as possible to its region. Symbols will usually remain quite small as a result. In the following we present an algorithm that computes high-quality necklace maps. We give several approaches to map the regions to their necklace intervals and we show how to maximize symbol sizes while keeping all symbols disjoint.

4 Computing Necklace Maps

Here we describe how to compute a necklace map with a single necklace. The extension to several necklaces is described in Section 5.

Definitions and notation. Before we can describe our algorithm we first need to introduce some definitions and notation. Our input consists of a set of polygons $\mathcal{P} = \{P_1, \ldots, P_n\}$ representing the geographic regions and a data set $\mathcal{Z} = \{z_1, \ldots, z_n\}$ where each value z_i ($z_i > 0$) is associated with polygon P_i. In the case of multi-polygon regions, such as the US or Indonesia, we either choose one representative polygon (contiguous US) or use the convex hull of all polygons. We assume that the data values are normalized, that is, $\sum_{i=1}^{n} z_i = 1$. We are also given the necklace: a star-shaped curve $\mathcal{C}$ with center $v = (x, y)$. For ease of explanation we assume the symbols S_i to be circles, our algorithm extends to other symbol shapes in a straightforward manner. We represent the location of the center of a circle S_i by an angle α_i relative to the center v of the necklace $\mathcal{C}$. The global scaling factor for all symbols is denoted by ρ. Since we use mathematical scaling, the radius of each symbol S_i in the final map is $\rho\sqrt{z_i}$.

For every polygon P_i we compute a single contiguous interval $I_i = [a_i, b_i]$ on the necklace with one of the methods described in the next subsection. The feasible interval I_i represents all acceptable angles to place S_i, counter-clockwise from a_i to b_i. We now say that a placement consisting of α_i ($1 \leq i \leq n$) and ρ is *feasible* if the following holds: (i) all S_i are disjoint, and (ii) $\alpha_i \in I_i$ for all $1 \leq i \leq n$. Our goal is hence to find a feasible placement that maximizes ρ. Given such a feasible placement we use a variety of approaches to improve the location of each symbol while keeping ρ fixed. Our algorithm can be summarized as follows:

Algorithm NecklaceMaps($\mathcal{P}, \mathcal{Z}, \mathcal{C}$)
1. compute feasible intervals *(see Section 4.1)*
2. optimize symbol sizes *(see Section 4.2)*
3. optimize symbol placements *(see Section 4.3)*

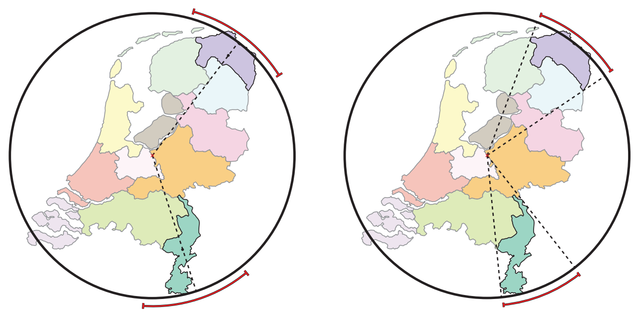

Fig. 5. Left: centroid intervals. Right: wedge intervals.

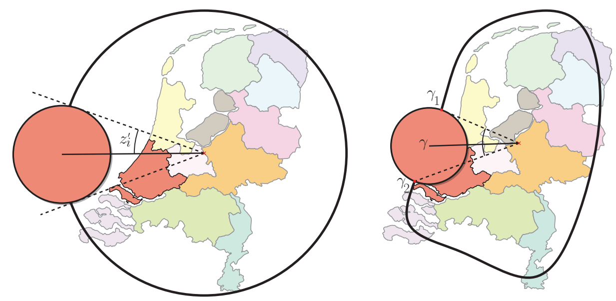

Fig. 7. Covering radius. Left: on circle. Right: on spline.

4.1 Computing feasible intervals

Depending on the type of input data and the input map different projections from regions onto necklace intervals can be suitable. Below we describe three approaches, the first two are implemented in our system and lead to very good results with various data sets and maps.

Centroid Intervals. Consider the ray from the center $v = (x, y)$ of the necklace $\mathcal{C}$ through the centroid (x_i, y_i) of a polygon P_i. The intersection between this ray and the necklace $\mathcal{C}$ at angle β_i can be considered a logical choice to place S_i. The feasible interval I_i is then defined as a constant-sized interval of size c around β_i (see Fig. 5 left). The value c can be a constant or depend on the number of regions n. Hence, using this method, the feasible intervals are computed as follows:

$$I_i = (\beta_i - \frac{c}{2}, \beta_i + \frac{c}{2}), \textit{ where } \beta_i = \textit{atan2}(y_i - y, x_i - x) \tag{1}$$

Note that with this method the length of all intervals is the same. However it is reasonable to assume that for larger regions, a larger interval is acceptable. If that is the case, then one should use one of the other two types of intervals.

Wedge Intervals. To strengthen the relation between the symbols and the regions, we can define the feasible intervals I_i to have a clear geometric meaning. We say that an angle α is acceptable for S_i if the ray from the center of $\mathcal{C}$ at angle α passes through P_i. Then I_i is the smallest interval containing all acceptable angles for S_i. In other words, I_i represents the smallest wedge of $\mathcal{C}$ containing P_i (see Fig. 5 right). Note that with this definition all angles are acceptable for the symbol S_j belonging to the polygon P_j that contains the center of $\mathcal{C}$. Since this is generally undesirable, we use the centroid interval for this particular symbol S_j.

With this method the relation between symbols and regions is relatively clear. However, wedge intervals do not take the data set $\mathcal{Z}$ into account. Wedge intervals of small regions are small, which can lead to sub-optimal maps if large data values are assigned to small regions. Hence we consider also density-dependent intervals.

Density-dependent Intervals. We can adapt the intervals depending on the data set $\mathcal{Z}$. Consider an interval of angles I. If an interval I_i is completely contained in I ($I_i \subseteq I$), then S_i must be placed in the interval I. We define the density of I as follows:

$$\delta(I) = \frac{1}{|I|} \sum_{i|I_i \subseteq I} \sqrt{z_i} \tag{2}$$

We enforce an upper bound on the maximum density $\max_I \delta(I) \leq c$, where c is a parameter that can be set by the user. For this, we change the sizes of the intervals iteratively. Whenever we find an interval I for which $\delta(I) > c$, we enlarge the intervals contained in I until $\delta(I) \leq c$ (see Fig. 6). We continue this process until $\max_I \delta(I) \leq c$. Note that the complete process requires us to check only $O(n^2)$ intervals. Of course, enlarging the intervals weakens the spatial relation between regions and symbols. The parameter c in fact controls the trade-off between symbol sizes and symbol positions. We can further improve this procedure by considering the fact that the outer two symbols placed in I are not completely contained in I. In fact half of these symbols can lie outside of I and should be excluded when computing $\delta(I)$.

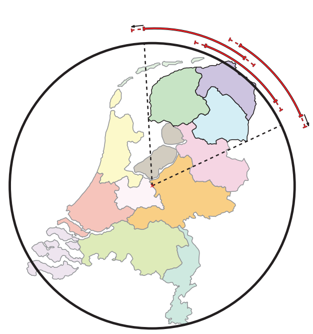

Fig. 6. Density-dependent intervals.

4.2 Optimizing symbol sizes

After computing the intervals, we can proceed with finding a feasible solution that maximizes ρ. For this we use a binary search on ρ. Hence, for a given ρ, we have to check if there is a feasible solution with that ρ. Because all symbols have to be placed on $\mathcal{C}$, this is essentially a 1-dimensional problem. However, we do need to know which part of $\mathcal{C}$ is covered by a symbol S_i with radius $\rho\sqrt{z_i}$. In case $\mathcal{C}$ is a circle with radius r, we compute the following.

$$z'_i = \textit{asin}(\frac{\rho\sqrt{z_i}}{r}) \tag{3}$$

It is easy to verify that a symbol with radius $\rho\sqrt{z_i}$ exactly covers a wedge of angle $2z'_i$ of $\mathcal{C}$, if $\mathcal{C}$ is a circle (see Fig. 7 left). We call z'_i the *covering radius* of S_i. If $\mathcal{C}$ is not a circle, computing the covering radius is somewhat more involved. If $\mathcal{C}$ is a circle, then the part of $\mathcal{C}$ that is covered by S_i is independent of the position of S_i. This is no longer the case if $\mathcal{C}$ is an ellipse or a cubic B-spline.

For a given position and size of a symbol S_i, we determine the part of the necklace covered by S_i, or the covering radius z'_i, as follows. Let S_i be at an angle γ from the center of the necklace. We trace from the center of S_i along the necklace in both directions. Let γ_1 and γ_2 be the angles where the necklace leaves S_i. Then the part of the necklace covered by S_i extends from γ_1 to γ_2. Since we cannot compute γ_1 and γ_2 exactly, we step through the necklace to approximate them. We can then define the covering radius at angle γ as $z'_i(\gamma) = \max(\gamma_2 - \gamma, \gamma - \gamma_1)$. Note that, using this definition, the wedge from γ_1 to γ_2 does not necessarily contain the symbol (see Fig. 7 right). Since S_i can be placed anywhere in its interval I_i, we use the largest covering radius for all possible placements of S_i. Thus, the covering radius of a symbol S_i is defined as follows.

$$z'_i = \max_{\gamma \in I_i} z'_i(\gamma) \tag{4}$$

Also in this case we step through a number of values of γ to approximate z'_i. Although this always over-estimates the part of the necklace

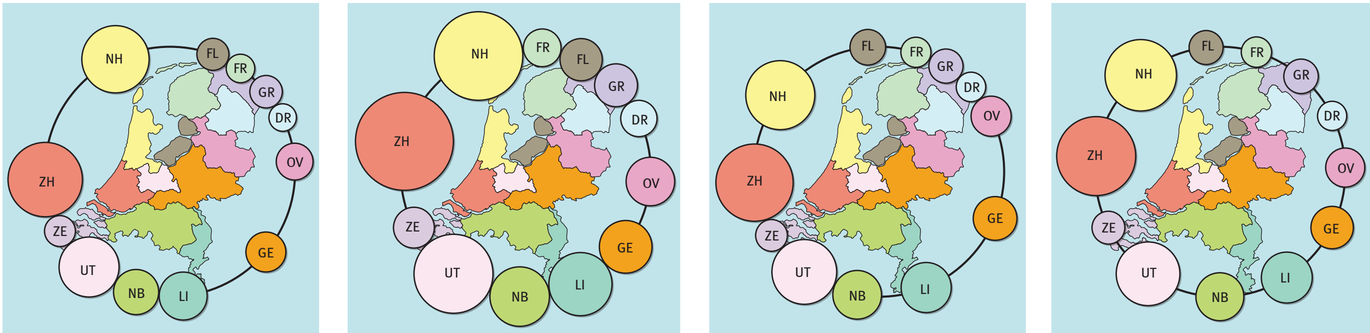

Fig. 8. Left: Optimized symbol sizes with centroid intervals. Center left: Optimized symbol sizes with wedge intervals. Center right: Symbols with buffers pushed towards middle. Right: Symbols with buffers pushed away from each other.

covered by a symbol, this error is limited as long as the necklace is fairly close to a circle. Alternatively, instead of using the precomputed z_i', we can compute the (exact) covering radius $z_i'(\gamma)$ when needed for an angle γ during the subsequent stages of our algorithm. This will give better results, but it makes our algorithm much more complicated and results in only a slight improvement. Hence we do not consider such an improvement in this paper.

With the covering radii, we can describe our problem as the following 1-dimensional problem. Find angles α_i ($1 \leq i \leq n$) such that:

- $\alpha_i \in I_i$ or $a_i \leq \alpha_i \leq b_i$ for $1 \leq i \leq n$
- The intervals $[\alpha_i - z_i', \alpha_i + z_i']$ are disjoint.

We can now consider two variants of the problem. We can either let the centers of the intervals I_i determine an order and require the symbols to follow this order (this would make sense with the centroid intervals), or we allow the symbols to be in any order. We make this distinction because there is a simple and fast algorithm for the first variant, but the second variant is NP-complete. However, the second variant is less restrictive, gives better results, and is hence more interesting. We have algorithms for both variants.

For the first variant, we can use the following simple approach. Initially, let $\rho = 0$ and place all symbols at the beginning of their interval. Then proceed by increasing ρ, growing the symbols. When two symbols hit, push them away from each other. Since all symbols start at the beginning of their interval, we can push in only one direction (counterclockwise). We continue this process until one symbol is pushed to the end of its interval. At this point we have found the optimal ρ and all symbols are placed on the necklace without overlap. More details can be found in our companion paper [15].

Unfortunately, the easy approach for the first variant does not work for the second variant. Instead we use dynamic programming to solve this variant. This results in a fixed-parameter tractable algorithm in the *thickness* of the intervals $I_1 \ldots I_n$. For a given α, let $k(\alpha)$ be the number of intervals I_i that contain α. We define the *thickness* K of a set of intervals to be the maximum of $k(\alpha)$ for all α. Note that, for wedge intervals, the thickness K has a clear geometric meaning: K is the maximum number of regions crossed by a single ray originating from the center of the necklace. The running time of our algorithm is $O(nK2^K)$. So far we have ignored the fact that the symbols have to be placed on a circle, not on a line. Extending our approach to a circle comes at some cost with respect to the running time, the details can be found in our companion paper [15]. The dynamic programming algorithm is quite flexible and allows us to impose a partial order on the symbols. Although the running time is still exponential, K will be rather small for most input instances ($K \leq 10$). Of course, we can easily construct an input instance that gives much larger K, but then the resulting necklace map would be quite unclear. In that case it is better to show fewer symbols or to use multiple necklaces. Figure 8 shows the necklace maps computed with the second variant with centroid intervals (left) and wedge intervals (center left). In fact, all necklace maps in this paper are computed with the second variant.

4.3 Optimizing symbol placements

With the algorithm described above, we get a feasible solution where the symbols are as large as possible. But, due to the way the algorithm works, the positions of the symbols are not optimal, even if we fix ρ. That is because the algorithm tends to place the symbols on the outer ends of their intervals, while it would be better to place the symbols in the middle of their intervals. It is also visually more appealing if the symbols are not packed together, but instead have some space between them. To remedy this problem, we use a post-optimization step to push the symbols to a better position. To not completely undo the work done by the placement algorithm, we do not change the sizes of the symbols and make only slight changes to the order of the symbols.

To push the symbols to a better position, we use a force-based method. Assuming that the order of the symbols is fixed, a symbol S_i at angle α_i has two neighboring symbols S_{i-1} and S_{i+1}. Let the distance between S_{i-1} and S_i be d_i and the distance between S_i and S_{i+1} be d_{i+1}. Here we mean the distance between the symbols in the 1-dimensional space, so $d_i = \alpha_i - z_i' - (\alpha_{i-1} + z_{i-1}')$. Let the middle of the interval I_i be m_i. Then we can define the following two forces acting on symbol S_i.

$$Frep(i) = f_r\left(\frac{1}{d_i} - \frac{1}{d_{i+1}}\right) \tag{5}$$

$$F_{mid}(i) = f_m(m_i - \alpha_i) \tag{6}$$

The constants f_r and f_m can be tuned to get a trade-off between pushing a symbol to the middle of its interval, and pushing the symbols away from each other. We can define the total force as $F(i) = Frep(i) + F_{mid}(i)$. The goal is to get $F(i) = 0$ for all i. We use a very basic approach to achieve this. We go several times through all symbols, and for each symbol S_i we assume that S_{i-1} and S_{i+1} are fixed. Then we solve the equation $F(i) = 0$. This is now a cubic polynomial in α_i, so it can easily be solved. This approach can be related to the Gauss-Seidel method to solve a linear system of equations. The algorithm quickly computes a good solution.

As mentioned above, we do allow slight changes to the order of the symbols. For every two neighboring symbols, we check if the two symbols can be swapped such that the result is still a feasible solution. We swap two symbols in such a way that $Frep(i)$ remains the same for all i. If the total force $F_{mid}(i-1) + F_{mid}(i)$ is smaller after the swap, then we swap the two symbols, otherwise we do not.

Using the force-based approach, the positions of the symbols can be optimized for a fixed ρ. However, since ρ is maximized, symbols could have little room to improve their position. To allow for a better trade-off between symbol size and symbol position, we allow the user to set a buffer for every symbol. This buffer is included when computing the optimal symbol size, so that afterwards every symbol has some empty space around it, corresponding to the buffer size. This space can be used to improve the positions of the symbols during this step of the algorithm. Figure 8 shows the resulting necklace maps after adding a buffer and pushing symbols towards the middle of their intervals (center right) or pushing symbols away from each other (right).

5 Extensions

In this section we discuss several extensions of our necklace map algorithm. In particular, we show how to handle multiple necklaces, either nested or disjoint, and how to add flows to a necklace map. Furthermore, we briefly mention some options for symbol shapes, for the display of multi-variate data, and for interactivity.

5.1 Multiple Necklaces

There are several reasons why it might not be sufficient to use only one necklace. First of all, we might want to display so many symbols, that one necklace would appear crowded and would force symbol sizes to be too small for effective communication. To that end we consider the *symbol coverage ratio*: the part of the map covered by symbols. On a good necklace map the symbols should cover a reasonable part of the complete map. Assume that we want to place n symbols on a circular necklace with radius 1. Further assume that all symbols have the same size. Optimally, every symbol covers a wedge of the necklace with angle $2\pi/n$. Using Equation 3, the radius of every symbol is $r = \sin(\pi/n)$. Since the area of the complete map is $(2+2r)^2$, the symbol coverage ratio λ_n for n symbols is described by:

$$\lambda_n = \frac{n\pi\sin^2(\pi/n)}{(2+2\sin(\pi/n))^2} = O(\sqrt{1/n}) \tag{7}$$

The symbol coverage ratio sinks below 30% if $n \geq 20$. Our experiments show that up to ≈ 20 symbols per necklace indeed allows for good symbol placement and sizing.

To display larger data sets the regions should be distributed over multiple necklaces. For example, consider a world map as the one in Fig. 14. It seems natural to use at least one necklace per continent. Another example is shown in Fig. 9 where the US states are distributed onto a northeast, midwest, south, and west necklace (according to the US census bureau regional division).

A second reason to use multiple necklaces are clustered data sets as the one depicted in Fig. 10. Here regions are clustered by the point in time in which the corresponding countries joined the EU. Since the EU grew "from the inside out" it seems natural to use nested necklaces for each cluster. Below we discuss in some detail how to create necklace maps with multiple disjoint or nested necklaces.

Disjoint necklaces. If we have k necklaces, the algorithm computes k optimal scaling values $\rho_1, \ldots, \rho_k$. To maintain the relative sizes, we have to choose $\rho = \min_i \rho_i$. Our algorithm ensures that symbols of the same necklace do not overlap, however, additional care must be taken to avoid overlap of symbols from different necklaces. A straightforward approach to remove such overlap is to binary search for the maximum scaling factor ρ such that all symbols are disjoint. To still be able produce necklace maps with sufficiently large symbol sizes, we have to ensure that necklaces are not too close together. Furthermore, when optimizing the symbol placements, we can let overlapping symbols exert forces on each other. Hence, when performing the binary search for the optimal ρ, we jointly optimize the placements for all symbols in every step of the search.

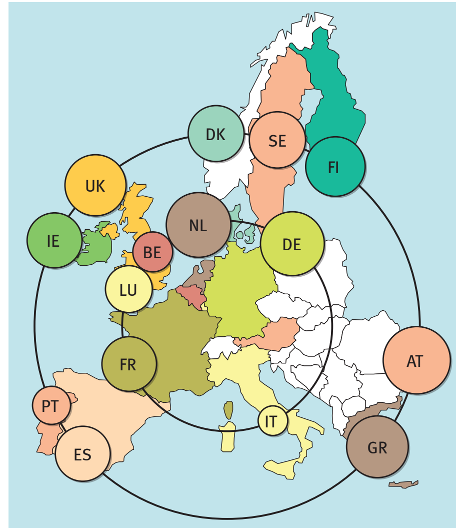

Fig. 10. Gender pay gap in 2007 as percentage of average gross hourly earnings of male paid employees. The inner necklace contains the first 6 EU countries, the outer necklace the additional 9 that joined by 1995.

Nested necklaces. If a necklace map has several necklaces then usually the user specifies the distribution of regions onto necklaces. However, there might be no obvious way to distribute the regions, but nevertheless, their number forces the use of several necklaces. In this case we propose the following simple approach to distribute regions onto nested necklaces. Intuitively, the innermost regions should be assigned to the innermost necklace and the assignment should grow outwards from that. We sort the regions based on increasing distance of their centroids from the center of the necklace and then greedily fill the necklaces from this list, taking into account the relative size of the symbols and the lengths of the necklaces. For best results the ratio between total symbol size and necklace length should be about equal for all necklaces. After this initial assignment we can consider a round of local swaps between necklaces, to facilitate larger symbol sizes and remove overlaps. As a final step we remove remaining overlaps by the same procedure as outlined above.

Note that grouping regions onto necklaces simply with the goal to maximize symbol sizes is somewhat dangerous. A necklace implies relationships between its regions. Hence one needs to be careful to not inadvertently associate regions that are not meant to form a group.

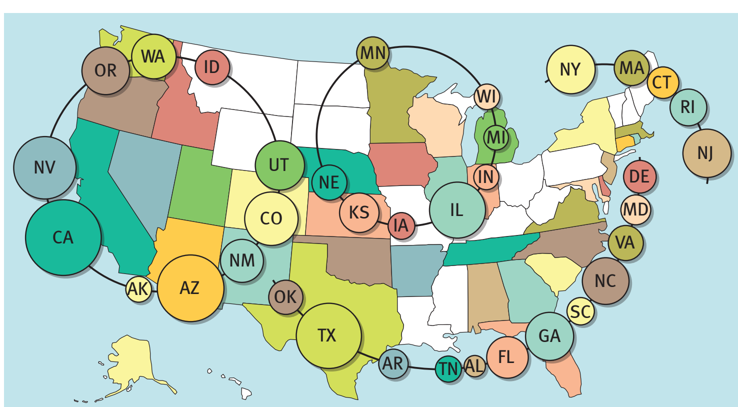

Fig. 9. Estimated number of illegal immigrants in 2000 in thousands, displaying only regions with a value of at least 0.5.

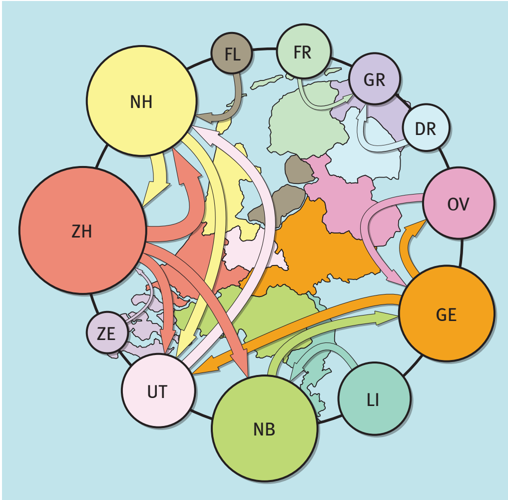

Fig. 11. Population of and relocation between the provinces of the Netherlands in 2005.

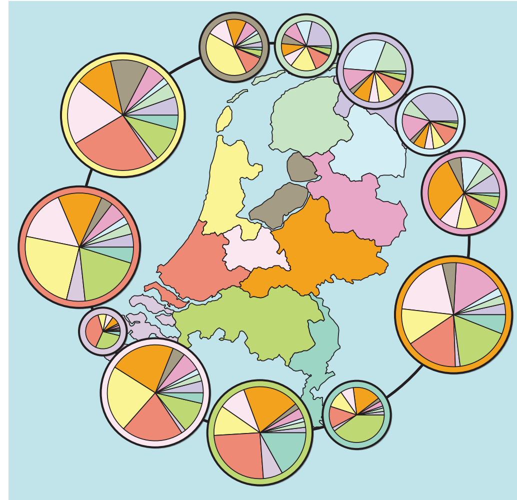

Fig. 12. Number of inhabitants moving to another province. The color of each "pizza slice" indicates the destination, the "pizza crust" the origin.

5.2 Flows

Necklace maps leave the interior of the map largely unoccupied. This allows us to visualize flows between regions as flows between symbols (see Fig. 11). As it is common in flow maps, we represent each flow by an arrow whose thickness corresponds to the amount of flow. In the following we describe how to attach multiple flows to one symbol and how to route flows in the interior of a necklace.

Consider a symbol S_i and the flows that need to be attached to S_i. The order of the symbols around the necklace induces an order for the flows. Note that symbol S_i can have both an incoming and an outgoing flow connecting to a symbol S_j, in this case the flows can be ordered arbitrarily. To attach the flows to S_i we use only a part of the boundary of S_i that is facing the center of the necklace. Using one fifth of the boundary works well in practice. To fit the flows onto this part of the boundary, we need to scale (the thickness of) the flows while maintaining their relative size. We do not allow incoming flows to overlap on the boundary of a symbol, but we do allow outgoing flows to do so. This does not only allow for thicker flows, but also gives a nice visual effect that is common in flow maps. Since we know the order of the flows, it is straightforward to compute the placement of the flows and the maximal scale factor ρ_i for the flows attached to a symbol S_i. The global scale factor for all flows is then $\rho_a = \min_i \rho_i$.

We describe a simple approach that gives good and natural looking results for convex necklaces and interior flows. Assume we need to draw a flow between symbol S_i at position $\vec{p_i}$ and symbol S_j at position $\vec{p_j}$, and that the flows are attached to the symbols at the positions $\vec{q_i}$ and $\vec{q_j}$. Let r_1 be the ray originating from $\vec{p_i}$ and passing through $\vec{q_i}$ and let r_2 be the ray originating from $\vec{p_j}$ and passing through $\vec{q_j}$. Now there are two possibilities: r_1 and r_2 do or do not cross. If r_1 and r_2 cross, say in $\vec{x}$, then we draw the flow as a quadratic Bezier curve with control points $\vec{q_i}$, $\vec{x}$ and $\vec{q_j}$. If r_1 and r_2 do not cross (or cross at a small angle), then let $d = \|\vec{q_i} - \vec{q_j}\|/4$. Then the first control point $\vec{x_1}$ is the point on r_1 at a distance d from $\vec{q_i}$. The second control point $\vec{x_2}$ is the point on r_2 at a distance d from $\vec{q_j}$. In this case, the flow is drawn as a cubic Bezier curve with control points $\vec{q_i}$, $\vec{x_1}$, $\vec{x_2}$ and $\vec{q_j}$. If we desire to leave the interior of the map unoccupied, we can use external flows. However, for nice looking flows that do not cross the necklace we need higher order Bezier curves. We can also add flows between necklaces, for example, to visualize trade between continents.

5.3 Symbol shapes, multivariate data, and interactivity

In principle any shape can be used for the symbols of a necklace map. However, symmetric symbols generally lead to better maps, since it is easier to estimate and compare sizes in this case (see Fig. 13 for an example using "ingots"). If the space that a symbol covers on the necklace is not invariant under rotation, then our algorithm does not necessarily produce symbols of maximum size.

Necklace maps can easily be augmented to display multivariate data. For example, consider a map that shows the work force per region divided into agricultural, governmental, and industry. In this case we can use a pie chart or histogram as symbol—a common mapping technique for human and economic geographic data. Such pie charts or histograms usually have the same size per region and need to be placed without overlap, which makes necklace maps a particularly attractive choice (see Fig. 12). Furthermore, in a necklace map the geographic information of the base map tends to remain clearly visible. Hence we can use hatching to mark the regions according to a second variable or use dot mapping for density information.

As mentioned before, the advantage of necklace maps come at a price: the association of a symbol with its region is weaker than with other types of maps. However, in an interactive setting there are simple approaches that can strengthen this association. Symbols and their regions can be highlighted whenever one or the other is chosen by the user. Furthermore, one could consider to continuously morph between a necklace map and a proportional symbol map for the same data set with the same symbol sizing. Since necklaces are star-shaped it is easy to compute a morph that maintains the mental map of the user.

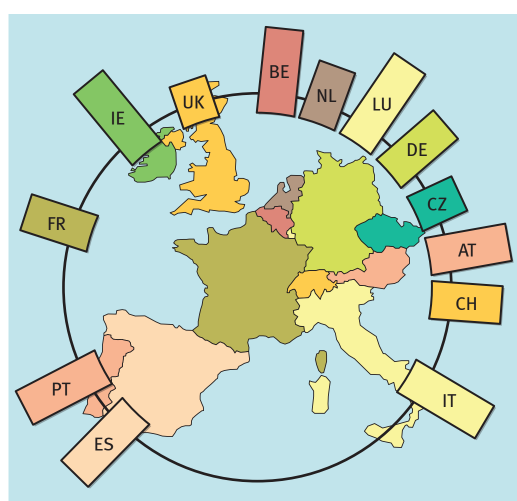

Fig. 13. Energy imports as percentage of energy consumption in 2007.

Fig. 14. FIFA World Cup 2010.

6 Application

We created a proof-of-concept implementation of the majority of our algorithms in Java. Our program supports centroid and wedge intervals, computes the optimal symbol sizes using the dynamic programming algorithm, and uses the force-based approach to optimize the symbol placements. It also implements the extensions described in Section 5. All figures in this paper were created by our program.

Our implementation is quite fast: all maps in this paper were computed in under a tenth of a second. Our implementation works for thickness $K \leq 15$. Maps with $K = 15$ take several minutes to compute; $K \leq 6$ for all maps in this paper. We have tested our implementation with up to 100 symbols on a single necklace, but we expect it to work for many more symbols, especially when using multiple necklaces. The thickness of geographic maps rarely exceeds 10 and if this is the case, then it is generally advisable to use several necklaces.

Our data stems from various sources: the European Commission Eurostat Program (`ec.europa.eu/ eurostat`), Worldmapper (`www.worldmapper.org`), the Swiss Federal Office of Energy (`www.bfe.admin.ch`), and Statistics Netherlands (`www.cbs.nl`). We use the ISO 3166-1 alpha-2 and alpha-3 two- and three-letter country codes to label our maps.

7 Discussion

The major advantage of necklace maps is their clear and uncluttered appearance. The linear ordering of the symbols along the necklaces makes it easy to estimate and compare symbol sizes correctly. Necklace maps visualize data sets well which are not proportional to region sizes and which do not contain data for all regions of the input map.

Our algorithm computes necklace maps of high quality: the relative spatial position of symbols and regions captures the spatial relation between regions and necklaces well and enables users to quickly associate a symbol with the correct region. Our symbols are disjoint and appear along the necklaces in an order which mirrors the neighbor relations of their regions. Furthermore, our optimization approach yields necklace maps whose symbols are as large as possible given the input.

There is a clear trade-off between symbols sizes and the spatial relation of symbols and regions which is also influenced by the number and distribution of symbols per necklace. Generally speaking it is better to use several necklaces instead of just one necklace. We are mapping two-dimensional data onto a one-dimensional domain. Intuitively, several necklaces increase the chances that a symbol can be placed close to its region and scaled to an appropriate size. In the extreme, many necklaces will bring us back to symbol maps. However, necklace maps with few necklaces still preserve their structured appearance while allowing for good symbol placement and sizing.

Clearly the shape and the exact location of the necklaces is very important for a good necklace map. Currently we create our necklaces by hand and add them to the input map. In the future we hope to be able to automatically generate suitable necklaces for given input maps and data sets, however, this seems to be a quite challenging algorithmic problem on its own. Also, our necklace maps currently lack a legend as it is commonly seen with symbols maps. Hence users can at the moment only judge relative sizes of symbols but cannot read absolute values off the maps. Finally, it would be interesting to explore animated necklace maps for time-varying data.

Acknowledgments

The authors wish to thank Marc van Kreveld and Jack van Wijk for helpful discussions on the topic of this paper, and Wouter Meulemans, Frank Razenberg, André van Renssen, and Jeaphianne van Rijn for help with the creation of the input maps. Bettina Speckmann and Kevin Verbeek are supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 639.022.707.

References

[1] M. A. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff. Boundary labeling: Models and efficient algorithms for rectangular maps. *Computational Geometry: Theory and Applications*, 36(3):215–236, 2007.

[2] S. Cabello, H. Haverkort, M. van Kreveld, and B. Speckmann. Algorithmic aspects of proportional symbol maps. *Algorithmica*, 2010. to appear.

[3] B. Dent. *Cartography - thematic map design*. McGraw-Hill, 5th edition, 1999.

[4] D. Dorling. *Area Cartograms: their Use and Creation*, volume 59 of *Concepts and Techniques in Modern Geography*. University of East Anglia, Environmental Publications, Norwich, 1996.

[5] J. A. Dougenik, N. R. Chrisman, and D. R. Niemeyer. An algorithm to construct continous area cartograms. *Professional Geographer*, 3:75–81, 1985.

[6] H. Edelsbrunner and E. Waupotitsch. A combinatorial approach to cartograms. *Computational Geometry: Theory and Applications*, 7:343–360, 1997.

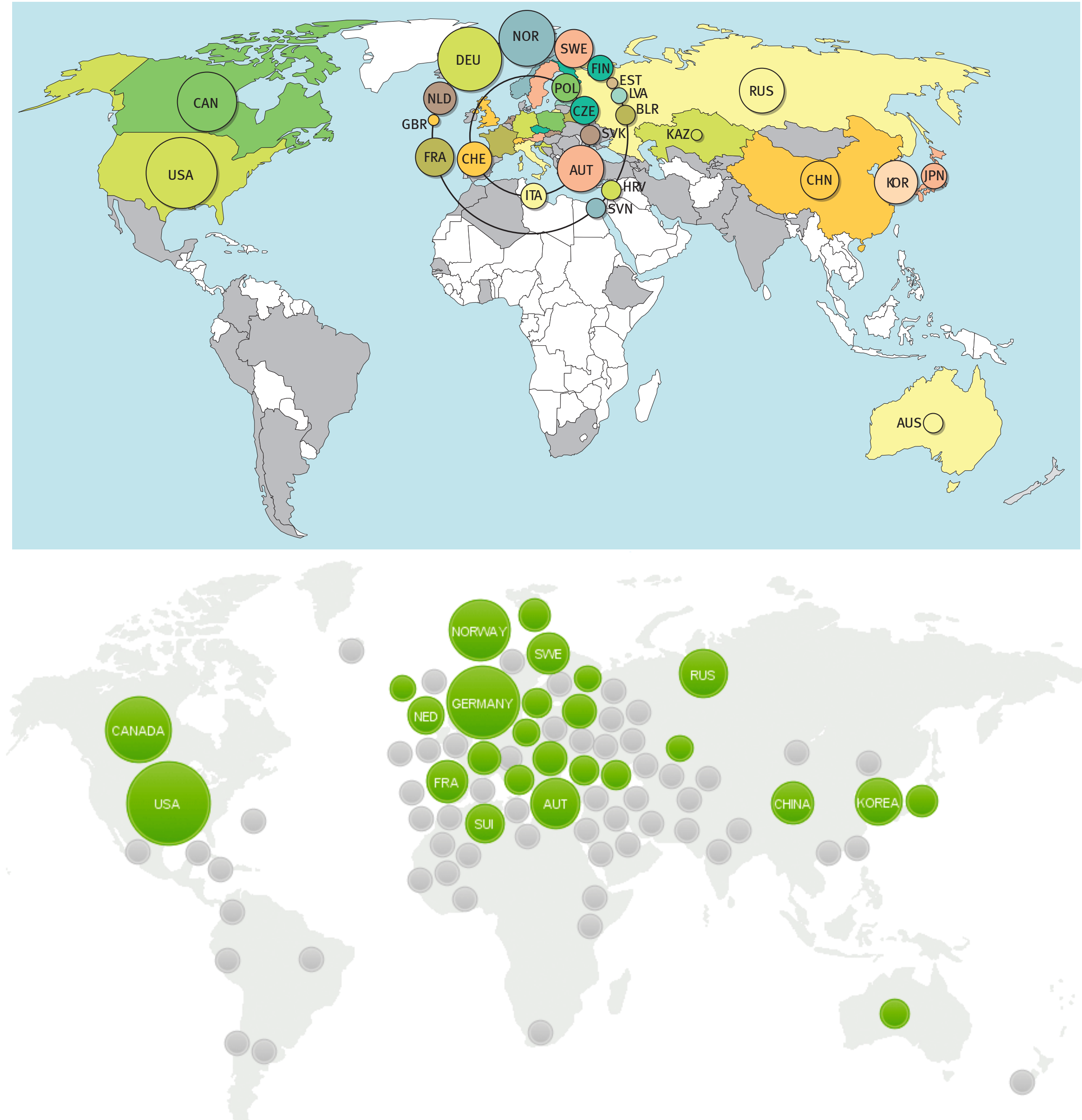

Fig. 15. Countries by total medals from the Vancouver 2010 Winter Olympics. Countries shaded gray or indicated by a gray dot did participate but did not win any medals. Above: a necklace map using several nested necklaces as well as individual symbols, countries are grouped by geographic location. Below: a screen shot from the geo view setting of the official olympic web-page `http://www.vancouver2010.com/olympic-medals/geo-view/`. The circle below Germany and above Switzerland depicts Italy.

[7] M. Gastner and M. Newman. Diffusion-based method for producing density-equalizing maps. *Proc. National Academy of Sciences of the United States of America (PNAS)*, 101(20):7499–7504, 2004.

[8] R. Heilmann, D. A. Keim, C. Panse, and M. Sips. Recmap: Rectangular map approximations. In *Proc. IEEE Symposium on Information Visualization (INFOVIS)*, pages 33–40, 2004.

[9] D. Keim, S. North, and C. Panse. Cartodraw: A fast algorithm for generating contiguous cartograms. *IEEE Transactions on Visualization and Computer Graphics*, 10:95–110, 2004.

[10] C. Kocmoud and D. House. A constraint-based approach to constructing continuous cartograms. In *Proc. Intern. Symposium on Spatial Data Handling (SDH)*, pages 236–246, 1998.

[11] M. v. Kreveld and B. Speckmann. On rectangular cartograms. *Computational Geometry: Theory and Applications*, 37(3):175187, 2007.

[12] E. Raisz. The rectangular statistical cartogram. *Geographical Review*, 24:292–296, 1934.

[13] T. A. Slocum, R. B. McMaster, F. C. Kessler, and H. H. Howard. *Thematic Cartography and Geographic Visualization*. Prentice Hall, 2nd edition, 2003.

[14] B. Speckmann, M. van Kreveld, and S. Florisson. A linear programming approach to rectangular cartograms. In *Proc. 12th Intern. Symposium on Spatial Data Handling (SDH)*, pages 527–546, 2006.

[15] B. Speckmann and K. Verbeek. Algorithms for necklace maps. In preparation, 2010.

[16] W. Tobler. Pseudo-cartograms. *The American Cartographer*, 13:43–50, 1986.

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

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