A Simple Pipeline for Coherent Grid Maps

A Simple Pipeline for Coherent Grid Maps

Wouter Meulemans, Max Sondag and Bettina Speckmann

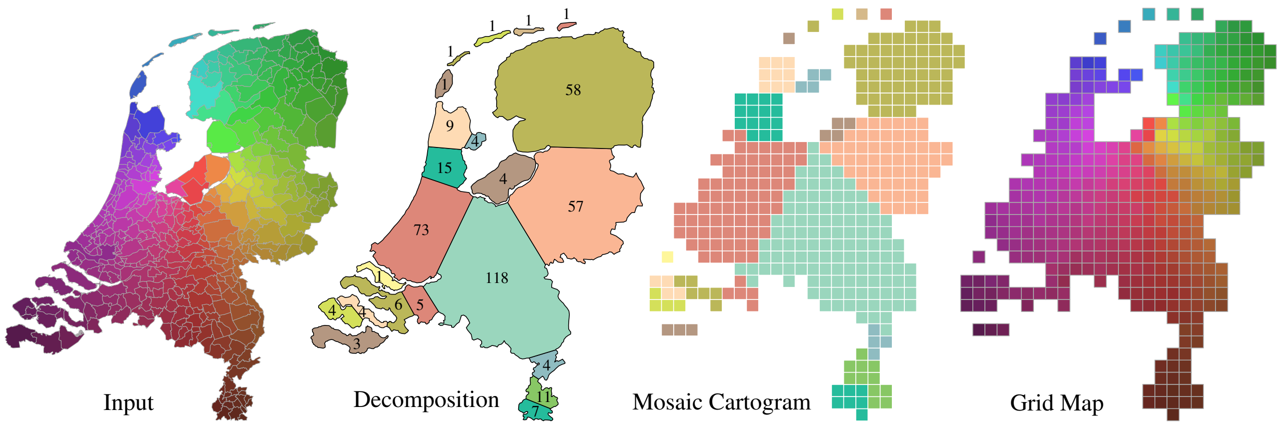

Fig. 1. Our 3-step pipeline to automatically compute a grid map, illustrated on the Dutch municipalities: (1) decompose the containing shape into parts; (2) compute a tile-based Mosaic Cartogram using these parts; (3) assign elements to tiles per part.

Abstract— Grid maps are spatial arrangements of simple tiles (often squares or hexagons), each of which represents a spatial element. They are an established, effective way to show complex data per spatial element, using visual encodings within each tile ranging from simple coloring to nested small-multiples visualizations. An effective grid map is coherent with the underlying geographic space: the tiles maintain the contiguity, neighborhoods and identifiability of the corresponding spatial elements, while the grid map as a whole maintains the global shape of the input. Of particular importance are salient local features of the global shape which need to be represented by tiles assigned to the appropriate spatial elements. State-of-the-art techniques can adequately deal only with simple cases, such as close-to-uniform spatial distributions or global shapes that have few characteristic features. We introduce a simple fully-automated 3-step pipeline for computing coherent grid maps. Each step is a well-studied problem: shape decomposition based on salient features, tile-based Mosaic Cartograms, and point-set matching. Our pipeline is a seamless composition of existing techniques for these problems and results in high-quality grid maps. We provide an implementation, demonstrate the efficacy of our approach on various complex datasets, and compare it to the state-of-the-art.

Index Terms—Grid maps, algorithms, tile maps, small multiples, geovisualization

1 Introduction

Data often has a spatial dimension which is an important factor when trying to understand relations and discrepancies between data at different locations. Correspondingly, many techniques exist to show data on a map, typically with high spatial accuracy – the traditional realm of thematic cartography. But as data complexity rises, maps that place information at the correct spatial location become unreadable, as visual elements necessarily become cluttered. Hence, some form of *schematization*, that is, controlled and deliberate distortion of the spatial dimension, is generally necessary to support visualizations of complex data while maintaining the spatial relations as well as possible.

One established and effective spatial schematization technique is the *grid map*. Primarily, each spatial element, such as a region or a site, is schematized into the same, simple *tile* – often a square, hexagon or other geometry that easily tiles the Euclidean plane. These tiles are then arranged in such a way as to reflect important characteristics of the spatial dimension, often using whitespace to capture salient local features. Grid maps are used to visualize geospatial data by popular news outlets [1,2,7,10,26,28,29,44], discussed by mapping enthusiasts and professionals [12,30–36,48,49], and applied in the academic literature [15,40,42,43,50,52].

The tiles of a grid map allow for visualizing data in a small-multiples style as popularized by Tufte [45]. Small-multiples are data-dense visualizations that juxtapose frames in a grid structure. Each frame displays the same visualization for different subsets of the data, for example, according to time steps or based on values in a categorical dimension. In a grid map, the spatial dimension determines both the subsets of the data as well as the arrangement of the juxtaposed frames. Guo et al. [15] present an intermediate variant, where space is linearized and used as one axis of a traditional small-multiples grid arrangement.

Small-multiple displays typically fill the available graphical space and arrange the frames according to their inherent order (time, data values, etc.). The spatial case presented by grid maps is inherently different. Viewers typically have a mental map of the geographic area and thus aspects such as recognizability and the ability to locate spatial elements based on expected location play a role. An effective grid map is *coherent* with the underlying geographic space: the tiles maintain properties such as contiguity, neighborhoods and identifiability of the corresponding spatial elements, while the grid map as a whole maintains the global shape of the input. Of particular importance are salient local features of the global shape which need to be represented by tiles assigned to the appropriate spatial elements.

• *Wouter Meulemans, Max Sondag and Bettina Speckmann are with TU Eindhoven. Email: {w.meulemans,m.f.m.sondag,b.speckmann}@tue.nl*

Manuscript received 30 Apr. 2020; revised 31 July 2020; accepted 14 Aug. 2020. Date of publication 7 Oct. 2020; date of current version 15 Jan. 2021. Digital Object Identifier no. 10.1109/TVCG.2020.3028953

Contributions and organization. In Section 2 we review related work and discuss the various facets of coherence in grid maps. As with any spatial deformation, perfectly coherent grid maps are generally impossible and one must make a trade-off between the facets. Computing a coherent grid map is hence a challenging multi-criteria optimization problem. However, the state-of-the-art shows that simple cases, such as close-to-uniform spatial distributions or global shapes that have few characteristic features, can be solved well, essentially using simple tile selection and assignment techniques (as long as sufficient care is taken to guarantee that connected input stays connected in the grid map). Our major contribution is the observation that any input can be decomposed into simple cases and that coherent solutions for the resulting simple subproblems can be combined into a coherent solution for the whole.

Based on this observation, we introduce a simple fully-automated pipeline to compute coherent grid maps in three steps; see Section 3. Each step is a well-studied problem: shape decomposition based on salient features (Section 4), tile-based Mosaic Cartograms (Section 5), and point-set matching (Section 6). Our pipeline is a seamless composition of existing techniques for these problems; we implement it in an open-source prototype[1]. In Section 7 we showcase the resulting high-quality grid maps, demonstrate the efficacy of our approach on various complex datasets, and compare it to the state-of-the-art.

2 Problem Exploration

Computing grid maps has two main components: ways to capture the various facets of coherence and algorithms to compute the actual grid maps. After exploring coherence in Section 2.1, we discuss algorithms in Section 2.2 by reviewing existing techniques and variations. But first we define the terminology we use throughout this paper.

Preliminaries. Our input consists of a set of *spatial elements*. These spatial elements can be either *sites* – specific point locations where data was obtained – or *regions* – typically, administrative boundaries such as countries or municipalities. In either case, we assume these to be contained in a *containing shape*. For sites, this is provided explicitly, as the (administrative) geographic shape that contains all sites. For regions, we typically assume that the regions partition the containing shape and thus it can be easily derived from the geometry of the regions. Typically, spatial elements are related through a *topology*, indicating pairs of adjacent spatial elements: elements that are connected or neighbors. For sites, these can be supplied explicitly (e.g., road connectivity between cities) or derived implicitly from their locations. For regions, the topology is implicitly represented through the shared boundaries of the geometry. We emphasize that our pipeline does not require a topology to be specified or derived, and a consideration of how such topologies can be derived is beyond the scope of this paper; we thus implicitly assume that a topology exists in the remainder of this paper.

[1]Available at `https://github.com/tue-aga/Gridmap`

We assume that a *tile* is a simple geometric shape that tiles the Euclidean plane. Typical shapes are squares or hexagons. A *tile arrangement* is a composition of a number of these tiles following the tiling of the plane; a tile arrangement selects a number of tiles from the infinite tiling. A *grid map* is a tile arrangement combined with an *assignment*: a one-to-one mapping between tiles and spatial elements.

2.1 Facets of coherence

There are myriad facets to coherence and these can be formalized in different ways; see e.g. the work by Meulemans et al. [24] for an extensive exposition. For the purpose of this paper, we categorize facets into *local* facets and *global* facets. The former represent facets of coherence with respect to individual spatial elements and their placement within a grid map; the latter capture facets of coherence with respect to groups of spatial elements and even aspects of the tile arrangement with respect to the containing shape. Based on existing work [11, 23, 24, 49], we identify the following facets (see Fig. 2).

Local distance [24]: distances between spatial elements should correlate to distances between their tiles. *Displacement* [11, 23, 24] is often used as proxy: the distance between a spatial element and its tile should be low. For example, in an overlay of Europe with a corresponding grid map, the distance between (the centroids of) each country and its representing tile should be small.

Local adjacency [11, 23, 24, 49]: spatial elements that are adjacent in the topology should be assigned to adjacent tiles in the grid map, and vice versa. For example, as Germany borders Denmark but not Italy, the tile for Germany should be adjacent to Denmark's tile but not to Italy's tile in a grid map of Europe.

Local direction [11, 23, 24, 49]: compass directions between spatial elements should match the directions between their tiles. For example, as the United Kingdom is north of Spain, the UK's tile should be above Spain's tile. Note that this is a local facet as it considers individual tiles, even though the spatial elements and their tiles need not be close in terms of distance. The global position [23] (referred to as location in [11]) of an element in the map can be seen as a variant of local direction.

Local shape: tiles on identifiable positions in the tile arrangement should be assigned to spatial elements that belong to that identifiable part in the containing shape. For example, Portugal should be the left-bottom most tile in a grid map of Europe; Florida should be the right-bottom most tile in a grid map of the US states. This

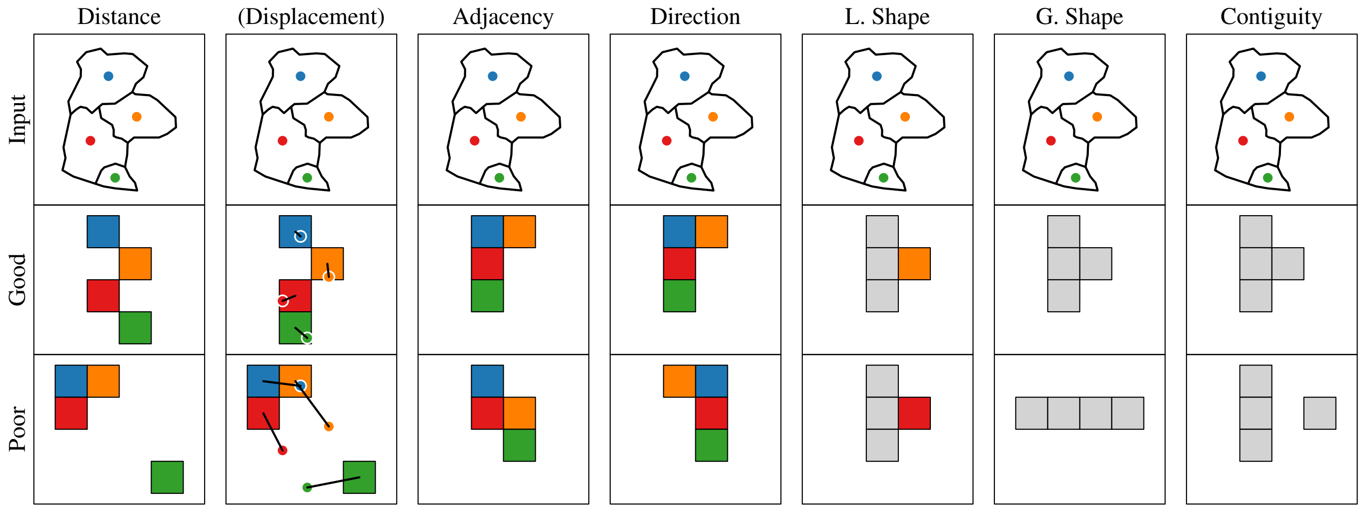

Fig. 2. Facets of coherence for grid maps. Displacement is not a facet in itself, but often used as proxy for distance and other facets.

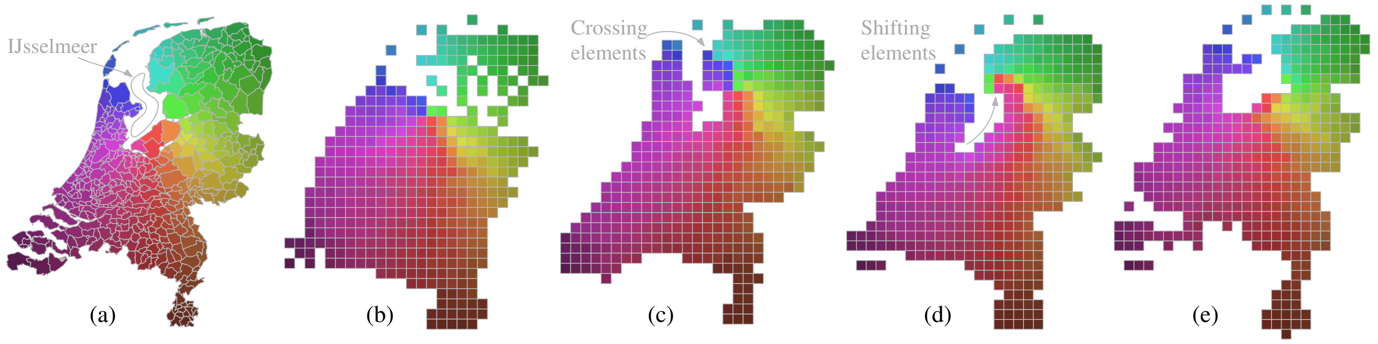

Fig. 3. (a) Dutch municipalities. (b) Tile arrangement which minimizes displacement is discontiguous. (c) Tile arrangement with good global shape, followed by displacement minimization. Note the municipalities crossing the characteristic gap in the center (the IJsselmeer) from west to east. (d) Tile arrangement optimized for displacement under the geodesic distance. Now the municipalities from the west shift around the IJsselmeer. (e) Our pipeline: mild deformation in the west to accommodate its municipalities, while maintaining the characteristic global shape.

facet is not explicitly mentioned in current work, but could be seen as a new interpretation combining the concepts of global position [11,23] with (global) shape. It is somewhat implicit in the "locate" task [11], though here the consideration of (nearly) full matrices avoids the need for shape in this facet.

Global distance, adjacency & direction: the containing shape may have different identifiable parts, due to recognizable features or a known subdivision of the containing shape. For example, Germany may be subdivided into its *Bundesländer* (states) for a grid map of its municipalities. The principles of local distance, adjacency and direction can be applied to these global parts. As generalizations of local variants, these are typically not explicitly mentioned in literature.

Global shape [23,24,49]: the overall appearance of the tile arrangement should resemble the containing shape. For example, a grid map of the Dutch municipalities should look like the Netherlands. Note that this facet is agnostic to the assignment of spatial elements to tiles.

Contiguity: this global facet captures topology on a coarser level. Specifically, contiguous parts of the containing shape should be represented by a contiguous set of tiles in the arrangement, and all and only spatial elements within the contiguous part should be assigned to this set of tiles. Note that contiguity does not require a topology to be given as it relies on the containing shape. For example, in a grid map of the US states, tiles for Alaska and Hawaii should not touch any other tiles; the tiles for the remaining contiguous states form one contiguous shape in the grid map. This criteria seems to be implicit in the considerations of Meulemans et al. [24] when discussing global shape.

Observe that satisfying all of the above facets is generally impossible. For example, not all adjacencies can be represented, distances will need to distort, and there may simply not be enough tiles to show all characteristic features well. Moreover, some facets readily conflict. For example, contiguity generally conflicts with direction and adjacency, adjacency in turn can conflict with direction, and shape (either local of global) can conflict with distance. Generally, we aim to optimize a trade-off between these facets. We consider contiguity to be a hard constraint, one that must be satisfied, as visual discontiguities are salient features, useful to identify e.g. islands. Unnecessary discontiguity thus has a high negative impact on local shape. Various facets can be captured through adequate measures, see [11,23,24,49]. However, global and local shape are particularly difficult to capture, while these facets are crucial to solving complex cases; see also our discussion Section 7.

2.2 Algorithms

The algorithmic problems arising from computing grid maps have been studied in various forms. Many specific formulations are computationally hard: for example, optimizing for even a simple version of local topology is NP-hard [5], and optimizing global shape under contiguity constraints is NP-hard as well, even to approximate [4,20]. Minimizing local directions with a given grid of tiles can be approximated [11], but its computational complexity is open.

Simple cases. If the tile arrangement for a grid map is a complete grid – the classic small-multiples setting – and the input has a close-to-uniform spatial distribution (for example, the majority of the *départements* of France have roughly the same area), then computing a grid map readily reduces to one-to-one point-set matching (of the region centroids to the grid) [11]. Given an alignment, minimizing the sum of Manhattan (L_1) or Euclidean distances can be solved efficiently [46], and an optimal alignment in terms of translation or scaling can be computed, though at significant computational cost [11]. The sum of squared Euclidean distances (L_2^2) can be minimized efficiently as well, including a simple optimal alignment in terms of translation [9]. The experiments by Eppstein et al. [11] show that optimizing L_2^2 also results in good local directions and adjacencies for a complete grid. This approach works well for other simple cases, for example, roughly convex containing shapes, such as the boroughs of London. These findings are corroborated by Meulemans et al. [24]: if the density of the regions is uniform, or the containing shape is mostly convex, or a mostly convex set of tiles is preselected, then aligning and minimizing displacement under L_2^2 yields coherent grid maps.

Complex cases. The situation is significantly more difficult when the containing shape has salient characteristic features, such as the IJs-

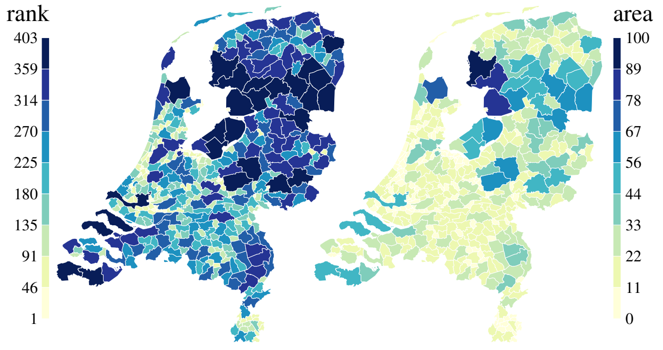

Fig. 4. Strongly varying region sizes in the Dutch municipalities. Regions are colored according to nine bins of equal quantiles (rank,left) or of equal area, indicated as a percentage of the largest region (area,right).

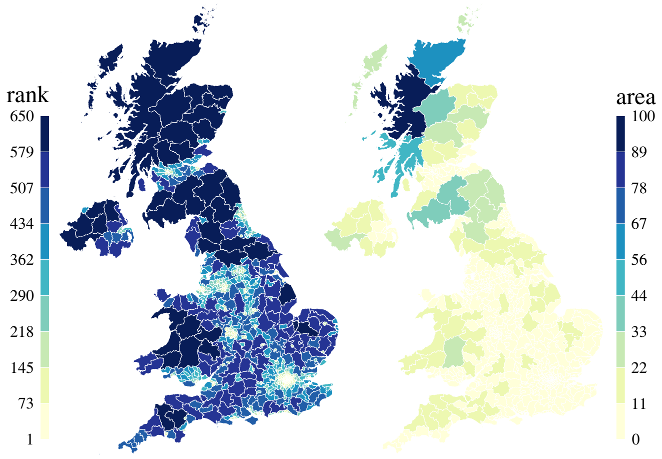

Fig. 5. Region sizes UK constituencies, color coding as in Fig. 4.

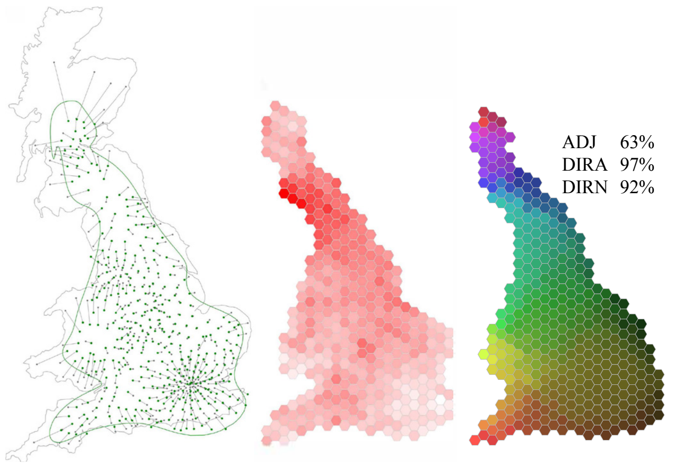

Fig. 6. UK local authorities. Left+Middle: Fig. 6 from [23]. Right: Reproduced and colored by authority locations. Scotland is eroded.

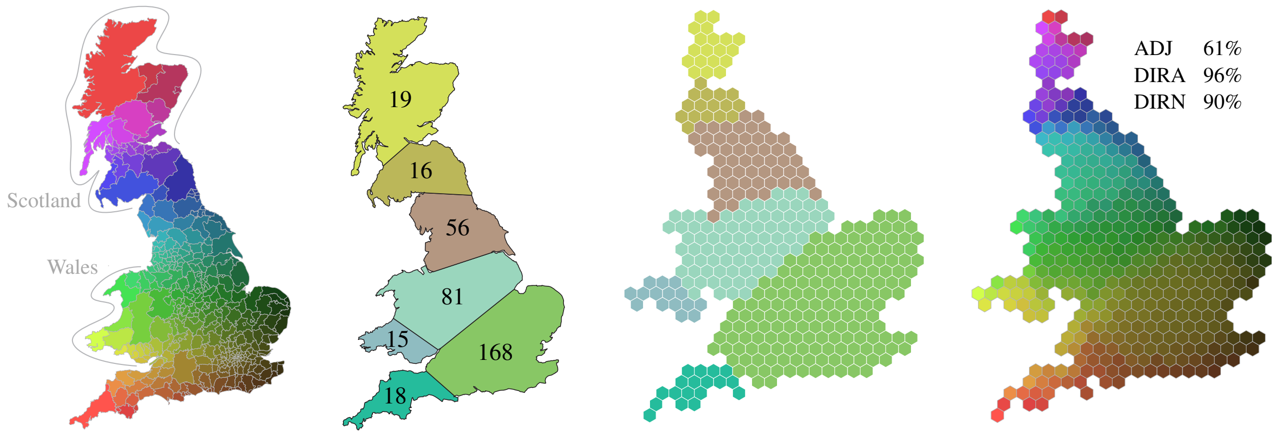

Fig. 7. UK local authorities using our pipeline: global and local shape are maintained.

selmeer in the Netherlands (see Fig. 3(a) – the large body of water in the middle), the Florida pan handle, or the west coast of the UK. A coherent grid map for such cases should use a tile arrangement that captures the global shape well. A priori, it is not clear how such a tile arrangement should be chosen from the infinite tiling. One could consider to simply select the tiles that subsequently allows for an assignment with the least displacement. Unfortunately, this approach generally leads to discontiguous tile arrangements: see Fig. 3(b). Meulemans et al. [24] show how to compute tile arrangements which make good use of whitespace (unselected tiles) and exhibit good global shape. However, if the region distribution is not uniform, then subsequent tile assignment while minimizing displacement under L_2^2 results in severe violations of local shape, see Fig. 3(c): municipalities from the densely populated west of the Netherlands travel across the IJsselmeer (municipality sizes as a proxy for density are shown in Fig. 4). Using a distance measure that is more aware of topology, such as the geodesic distance, does not alleviate the problem: see Fig. 3(d) where the tile assignment is optimized for the geodesic distance and as a result the municipalities now shift around the IJsselmeer.

As the above illustrates, the major challenge to resolve is the combination of varying density of spatial elements with characteristic features of the containing shape. Note that this combination is certainly not unique to the Netherlands and is also present e.g. in the UK constituencies: see Fig. 5 for a density overview.

McNeill and Hale [23]. To the best of our knowledge, McNeill and Hale [23] present the only automated method that aims to explicitly address complex cases, that is, the varying densities of spatial elements. Their method translates the spatial elements (regions only) and their topology into a graph. To obtain uniform distances between neighbors, they employ an approach that uses a form of spring-embedder: vertices are attached by springs to both their neighbors in the graph and to the centroids of their corresponding regions. The containing shape is deformed along with the vertices; the authors find a scale factor such that the deformed shape contains exactly the right number of tiles. Finally, the regions are assigned to tiles using a point-set matching algorithm to minimizes displacement under L_2^2.

We identify two major drawbacks of their method. (1) The tile arrangement is not necessarily contiguous, even if the input is contiguous. Their method ensures only that the scaled deformed shape contains the correct number of tiles, but not that these tiles are actually connected. See, for example, the five brown regions in Fig. 8, from an online implementation[2] by McNeill. (2) Characteristic features of the containing shape are eroded if parts need to grow or shrink considerably due to varying density, thereby reducing global and local shape. See, for example, Fig. 6 which shows the local authorities in the UK [3]: Scotland has been eroded and Wales lost most of its characteristic coast line. In comparison, our pipeline (Fig. 7) preserves global and local

[2]`https://observablehq.com/@gjmcn/make-a-tile-map`, 28.04.2020

[3]The mapping from authorities to cells could not be reproduced after correspondence with the authors due to non-deterministic behavior. Fig. 6 (right) is the closest match generated by their demo tool `https://gjmcn.github.io/tile-maps-eurovis-demo/`, 28.07.2020

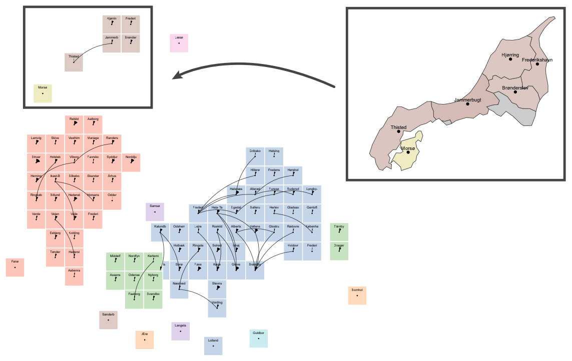
Fig. 8. Output of [23]: municipalities of Denmark; the five brown contiguous regions of Nørrejyske Ø (inset) map to discontiguous tiles.[2]

shape. McNeill and Hale focus on achieving a uniform distribution – a "simple case" – but they do so for the complete containing shape at once and hence lose control (to some degree) over contiguity and shape. In contrast, our pipeline uses decomposition into simple containing shapes to arrive at simple cases, which offers us more control over global and local shape and allows us to guarantee contiguity.

Cartograms. In some sense a grid map is a cartogram, where every spatial element (region) has unit weight. There are many cartogram techniques which focus on optimizing e.g. adjacencies, aspect ratio and cartographic error (difference between the target size of a region and the actual size in the output). However, most cartograms are neither aligned on a grid nor able to produce the desired tile shape. One could of course consider to rasterize the output of the density-equalizing cartogram algorithms proposed by Gastner and Newman [13] and by Gastner et al. [14]. However, it will generally be impossible to guarantee contiguity, as smaller regions tend to become rather elongated. Spatially Ordered Treemaps [51] combine ideas of tree maps and cartograms to show hierarchical data; also non-space-filling variants exist [41]. However, such tree maps do not align with tiles on a grid.

One type of cartogram, however, is in principle well suited for the computation of grid maps: the *Mosaic Cartograms* by Cano et al. [6] represent regions with multiple tiles, where the number of tiles corresponds to (integer) data values associated with each region. As such, Mosaic Cartograms cannot directly be used to compute coherent grid maps, but they are an integral part of our pipeline.

Semi-automatic. Wongsuphasawat [48] describes a semi-automatic pipeline to create grid maps. The first step computes a coarse, discontiguous grid map, using overlap removal and grid snapping. The result has significant discontiguities, which are then patched in a second step, by manually shifting parts of the arrangement until the result is contiguous and has a suitable global shape.

3 A 3-Step Pipeline for Coherent Grid Maps

Our discussion in the previous section points directly towards a natural pipeline to compute coherent grid maps: decompose the problem into one or more simple cases and then apply point-set matching. The simple case our pipeline achieves is arguably the simplest – that of convex containing shapes: we decompose the containing shape into relatively simple, mostly convex parts. Then we compute a Mosaic Cartogram [6] based on the parts and the number of spatial elements in each part. The Mosaic Cartogram grows or shrinks each part accordingly, while ensuring contiguity and maintaining shape: the tile arrangement per part is thus mostly convex as well. Point-set matching for the tile arrangement of each part completes the pipeline. Concretely, we use the following three steps, each of which is discussed in detail in the subsequent sections (see also Fig. 1).

Step 1 – Decompose into parts: we decompose the containing shape based on salient features into simple, mostly convex parts and note how many and which spatial elements each part contains.

Step 2 – Arrange tiles: we compute a Mosaic Cartogram, based on the parts, using the number of spatial elements as weight. The Mosaic Cartogram by design maintains shape, adjacencies and directions, and also ensures contiguity. The result is a tile arrangement where each tile is associated with one part and each part is represented by the correct number of tiles.

Step 3 – Assign elements: per part, we use point-set matching to compute an assignment that minimizes the sum of squared Euclidean distances between the spatial elements and the tiles.

Each step of the pipeline is thus a well-studied problem. In principle, different techniques could be used for each step – especially for Step 1 – though the literature suggests that our proposed composition is effective. To seamlessly create the complete pipeline, we made small changes to existing techniques or simplified existing approaches.

4 Decompose Shape Based on Salient Features

The first step of the pipeline decomposes the containing shape into parts, based on salient features. We can treat each contiguous part of the input separately, and hence can restrict our attention to the decomposition of a simple polygon P. Our output are the parts of P together with the spatial elements contained in each part.

Related work. Shape decomposition is an extremely well-studied problem. Below we sketch some of the most important considerations and approaches. An important related field is that of object recognition, which considers many aspects of objects such as shape, color, texture, motion, context, etc. It has been observed, though, that humans can often recognize an object solely by its shape [17], based on a decomposition into visually salient parts [17, 27, 37–39], via so-called *cuts*: straight boundary-to-boundary line segments fully within the shape.

Based on psycho-physical findings, a number of rules and constraints have been proposed in the literature to mimic the human visual decomposition in finding cuts. Though more general, we present them here for the case of simple polygons, as this is the setting for our pipeline. The most recognized of these rules are the minima rule [17] (a cut should always start at a reflex vertex) and the short-cut rule [39] (shorter cuts are preferred over longer cuts if they are otherwise equivalent).

To determine candidate cuts, the definition of a part-cut [38] is often used which stipulates that a cut should cross a local axis of symmetry, in addition to being contained within the shape and following the minima rule. Local symmetry can be seen as a weak form of symmetry: the two sides along each axis of local symmetry are shape-wise similar, but are not strictly symmetric. The medial axis [3] is often used for this purpose (e.g. [27]), but different formulations are possible.

Typically, there are many candidate cuts, and thus there is a need to filter and select an appropriate set of (final) cuts, see for example [18, 21, 25, 27]. Our implementation is inspired by the work of Papanelopoulos et al. [27] who use the medial axis to construct a candidate set, which is hence based on local symmetry.

Our implementation. Using the medial axis to identify candidate cuts guarantees that all cuts are pairwise disjoint. Hence, any combination of candidate cuts yields a valid decomposition into parts. Specifically, let $\mu(P)$ denote the medial axis of our input polygon P and consider a single segment s of $\mu(P)$. Note that $\mu(P)$ consists of both straight line

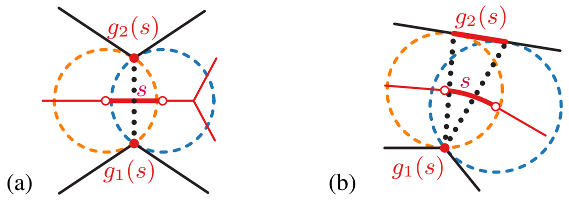

Fig. 9. Candidate cuts (dotted lines): medial axis segment s is (a) a straight segment, or (b) a parabolic arc. Inscribed circles (blue and orange) centered on endpoints of s define line segments $g_1(s)$ and $g_2(s)$.

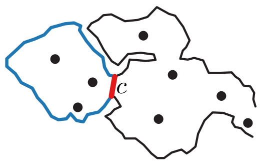

Fig. 10. Candidate cut c with dilation 14.5 and productivity 3. Shortest length along the boundary in blue; dots represent spatial elements.

edges and parabolic arcs. Let $g_1(s)$ and $g_2(s)$ denote the two shortest distinct line segments along the boundary of P, such that all inscribed circles of P with their center on s touch both $g_1(s)$ and $g_2(s)$. Note that either one or both of $g_1(s)$ or $g_2(s)$ can be a reflex vertex. If both are reflex vertices, then we add a single candidate cut between them (see Fig. 9(a)). If one is a reflex vertex, then the other is a line segment and we add two candidate cuts (see Fig. 9(b)): from the reflex vertex to the two endpoints of the line segment. Since at least one endpoint of each cut is a reflex vertex, all candidate cuts follow the minima rule.

Our filtering scheme uses a dilation threshold d and a productivity threshold p (see Fig. 10). The dilation (or detour factor) of a cut is the shorter of the two lengths along the boundary of P divided by its Euclidean length. This ratio is always at least 1 due to triangle inequality. The productivity of a cut is a new concept, tailored to the construction of grid maps. Each cut partitions (the centroids of) the spatial elements into two sets. The productivity of a cut is the number of elements in the smaller set. Our pipeline does not allow cuts with low productivity to ensure that each part has sufficient tiles at its disposal to achieve good local and global shape.

We select candidate cuts as follows. First, we sort them by increasing length following the short-cut rule and then process cuts one-by-one. If the dilation of a cut is above d and its productivity is above p, then we apply the cut: we partition P and its spatial elements and recurse on the resulting two subpolygons. In this recursive step, we update both dilation and productivity of each candidate cut with respect to its new subpolygon. See Fig. 11(b) for an example of the eventual result.

Let n denote the complexity of the containing shape and m the number of spatial elements. As the medial axis takes $O(n)$ time to compute and has linear complexity [8], we compute our $O(n)$ candidate cuts and sort them in $O(n \log n)$ time. Per candidate cut we keep track of the length along the (sub)polygon boundary and the number of elements on both sides of the cut, from which we can derive dilation and productivity in $O(1)$ time. Initializing these values takes $O(n^2 + nm)$ time in total, using a straightforward implementation. When a candidate cut is selected, we can update these values in $O(1)$ time each. We leverage the medial-axis structure to traverse the candidates on both sides and update all values in $O(n)$ total time. As such, the selection process can be executed in $O(n^2)$ time. The bottleneck is hence initializing the selection process in $O(n^2 + nm)$ time.

Using other methods. Our specific implementation can easily be replaced by other algorithms that detect cuts based on salient features. However, for the next step in the pipeline, it is important to avoid cuts with low or even zero productivity. A part with zero spatial elements cannot be represented by a Mosaic Cartogram and will hence violate topology and thereby contiguity.

5 Arrange Tiles Using Mosaic Cartograms

The second step of our pipeline uses the parts – the decomposition result – to compute a tile arrangement. We do so for all parts simultaneously, to ensure that tiles do not overlap, to ensure contiguity, and to optimize other global facets of coherence. The inputs for this step are the parts, that is, a subdivision of the simple polygon P, together with the number of spatial elements that each part represents. The output of this step is a Mosaic Cartogram: a contiguous tile arrangement, including a mapping from parts to contiguous sets of tiles of the correct cardinality.

To the best of our knowledge, the original algorithm for Mosaic Cartograms [6] is currently the only method to solve the given problem. However, other algorithms could in principle replace this technique, as long as they can guarantee contiguity and represent each part with the exact number of tiles – see also Section 7.

Our implementation. We use Mosaic Cartograms [6], which deform in a shape-aware manner, while maintaining contiguity and global adjacencies between the parts, while optimizing for global directions between neighboring parts; see Fig. 11(c) for an example result.

A Mosaic Cartogram is roughly computed as follows (refer to [6] for details). First, an initial tile arrangement is computed, such that each part has the correct adjacencies, without accounting for shape or the necessary number of tiles. Then, for each part, the algorithm computes a guiding shape: a contiguous set of tiles that resembles the shape of the part. The tile arrangement is then iteratively modified, by moving guiding shapes and changing tiles, while ensuring that adjacencies are maintained. Movement is based on the desired direction between two parts, and tiles are changed to also converge on the required number. This process stops when the guiding shapes no longer move and tiles no longer change. At this point the exact number of tiles per part might not yet be achieved, but typically it is close. A final post-processing step corrects the number of tiles and fills any unwarranted gaps in the tile arrangement, while still maintaining the correct adjacencies.

The correct number of tiles and correct adjacencies might not be compatible (e.g., a part with only a single tile that needs to be adjacent to many other parts). Indeed, this can be expected with a low productivity threshold in the first step, though parts with few spatial elements typically have few adjacent parts. Whereas Mosaic Cartograms opt for a slight deviation in the number of tiles (cartographic error) to ensure

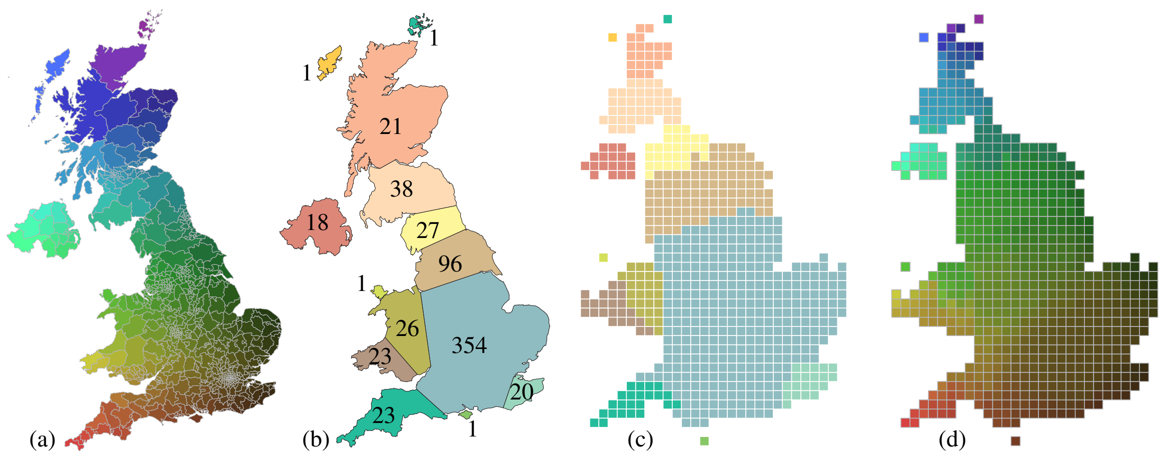

Fig. 11. UK constituencies, computed with dilation $d = 3$ and productivity $p = 20$. Numbers in (b) indicate number of spatial elements per part.

the right adjacencies between parts, this is not an option in our pipeline: each part must be represented by as many tiles as the number of spatial element it contains. If every part is adjacent to the "outside" (i.e., the adjacency graph is outerplanar), the post-processing step has significant freedom, and typically sufficiently so to achieve the right number of tiles. This is the case in all our test instances; without subdivisions (see Section 7) the topology is even a tree, by design of our decomposition step. We therefore did not observe any deviation from the exact number of tiles, even while perfectly representing adjacencies of the parts.

If needed, one could add a second post-processing step, similar to the first; but rather than requiring that adjacencies are preserved, we enforce only that the result remains contiguous. In this way we can ensure that the resulting Mosaic Cartogram has zero cartographic error and is contiguous, at the possible cost of small topological violations. It is an interesting direction for future work to compute good Mosaic Cartograms with zero cartographic error more directly.

The algorithm for Mosaic Cartograms assumes a contiguous input. We generate such a cartogram for each contiguous piece of our input and compose them. Our implementation features some automation, based on direction. Specifically, we start with the largest piece (for example, mainland UK or mainland NL) and simply use its result. For any subsequent piece (usually islands of decreasing size), we use the direction between the closest spatial element of the new piece towards any of the already-placed pieces. This gives a starting tile to place the new piece, from which we search locally for a placement that does not cause overlap or new adjacencies. This automation works reasonably in simple situations (e.g., UK: mainland with six well distributed islands) but requires manual post-processing for maps with more intricate layout (e.g., Netherlands: mainland with seven clustered islands). We apply this optional manual step in all our results, and leave better placement of separate pieces in Mosaic Cartograms to future work.

6 Assign elements to tiles via point-set matching

All that remains now, is to assign spatial elements to tiles in the tile arrangement computed in Step 2. As the sets of tiles and the number of spatial elements per part match exactly, we can execute this step separately for each part. The input is thus a set of contiguous tiles for one (mostly convex, and hence simple) part, together with the spatial elements for that part. The output is the tile assignment for this subset of the spatial elements. Combining all assignments yields the grid map.

Related work. There are various ways to assign spatial elements to a tile arrangement. For tile arrangements (close to) a grid, one can use a greedy assignment as proposed by Wood and Dykes [51] for Spatially Ordered Treemaps and adapted by Eppstein et al. [11] for grid maps (called *SpatialGrid* in [11]). Eppstein et al. [11] propose a variety of other methods for this case, including displacement optimization under the sum of squared Euclidean distances (L_2^2). They show that optimizing L_2^2 also results in good local directions and adjacencies; these findings are corroborated by Meulemans et al. [24]. Since we can minimize L_2^2 efficiently [9], this approach is generally the method of choice and also the one we employ in our pipeline. Later work by Liu et al. [19] proposes a form of constrained multi-dimensional scaling to assign regions to a grid. However, in addition to the restriction to the grid, the assignment quality is generally lower than the one via L_2^2.

Our implementation. As stated above, we optimize for L_2^2, which yields good results in local facets of coherence for the simple cases that the earlier steps in the pipeline create. To align the spatial elements and the set of tiles of a part, we compute the affine transformation such that the bounding box of the set of tiles is equal to that of the centroids of the spatial elements. Subsequently, we use point-set matching from centroid to centroid to minimize the sum of squared Euclidean distances in this transformed space, implemented as a linear program [9].

7 Results and discussion

Here we investigate the efficacy of our pipeline in three aspects: (1) parametrization of our shape-decomposition step; (2) a comparison to state-of-the art; (3) using a known subdivision of the containing shape. We end this section with a general discussion and future work.

Parametrization. Contrasting the other two steps in our pipeline, the first step of decomposing the containing shape is parametrized. Setting suitable thresholds is important to achieve the best results. Here we briefly investigate the effect of these parameters via the result of the subsequent step: the tile arrangement.

Fig. 12 shows results for the Dutch municipalities, using a dilation threshold $d \in \{2,3,5\}$ and productivity $p \in \{4,10,30\}$. The effect of d is quite straightforward: a lower value of d means that more candidate cuts can be selected and thus results in more identified parts. Productivity p has a similar effect: parts are required to contain more spatial elements and thus higher values lead to fewer parts. With too many parts, the computed Mosaic Cartogram is too detailed, resulting in a jagged, uneven appearance in places where this is not necessary for global shape – this is because Mosaic Cartograms consider shape purely on a part-level. Yet, with too few parts we do not obtain the simple case within each part due to uneven distributions of the spatial elements while still having recognizable features. Hence, both need to work together to obtain parts that represent simple cases and that are both visually salient as well has having enough elements to be represented reasonably. Of these provided figures, we deem that the case of ($d = 3, p = 4$) strikes this balance well and nicely places southern Flevoland (four municipalities in the central brown region in the figure) along the IJsselmeer. We use these settings and hence this Mosaic Cartogram also in Figs. 1, 3(e) and 14(c) showing the Dutch municipalities.

Productivity p can likely be configured similarly across instances of similar size, though effects of geography can be expected as culturally significant features bias towards certain decompositions. However, dilation threshold d faces a difficulty arising from a well-known geographic phenomenon: the coastline paradox [22]. In our situation, the detail at which the containing shape is represented immediately and potentially greatly affects the length along the polygon boundary, used to determine dilation; yet, the length of a candidate cut is largely unaffected. As a result, the dilation threshold for a particular instance depends on the resolution of the input map. A normalization of the shape boundary is not quite enough, as shape complexity remains a factor and borders may locally vary in semantics and thus in their effect on the local boundary length – consider, e.g., differences between actual coastlines and borders with other countries such as Norway with its fjords and straight border with Sweden. A suitably normalizing simplification of the containing shape might provide a solution to this resolution-dependence; we leave this investigation to future work.

Comparison. With Figs. 6 and 7 shown earlier, we compare our method to the approach of McNeill and Hale [23], using a hexagonal grid map for the UK local authority districts. These figures clearly show that our pipeline retains more of the characteristic features, especially around Scotland and Wales. The smooth gradient of colors in our result also shows that local facets are preserved well. To further compare the two methods on the local facets of coherence, we use the measures proposed by Meulemans et al. [24] for local adjacency and local direction: The percentage of adjacencies maintained (ADJ), the percentage of orthogonal directions maintained (DIRA), and the percentage of orthogonal directions between neighbours maintained (DIRN). The two methods have comparable scores for the UK map. We also provide a grid map with square tiles in Fig. 11 on the UK constituencies, showing that this improved shape is also achieved with other tile shapes.

Fig. 13(d,h) shows the result of our pipeline on the contiguous states of the US without and with Washington DC, a common example and use case of grid maps. We observe that this case is not necessarily complex, as evidenced by the few parts in the decomposition (Fig. 13(b,f)), but not quite simple either: e.g., Florida and the Great Lakes give characteristic features for local shape that can be preserved, and states along the (north)eastern coast are significantly smaller than the other states. That only few parts are identified signals that our pipeline can adequately cope with simpler cases as well, without the earlier steps in the pipeline needlessly distorting the eventual result. That is, it can be applied nearly agnostic to the input, beyond the configuration of the dilation and productivity thresholds as discussed above. For comparison with Fig. 13(h), a result of McNeill and Hale [23] is shown in Fig. 13(e) using our color scheme. The maps are roughly of comparable quality,

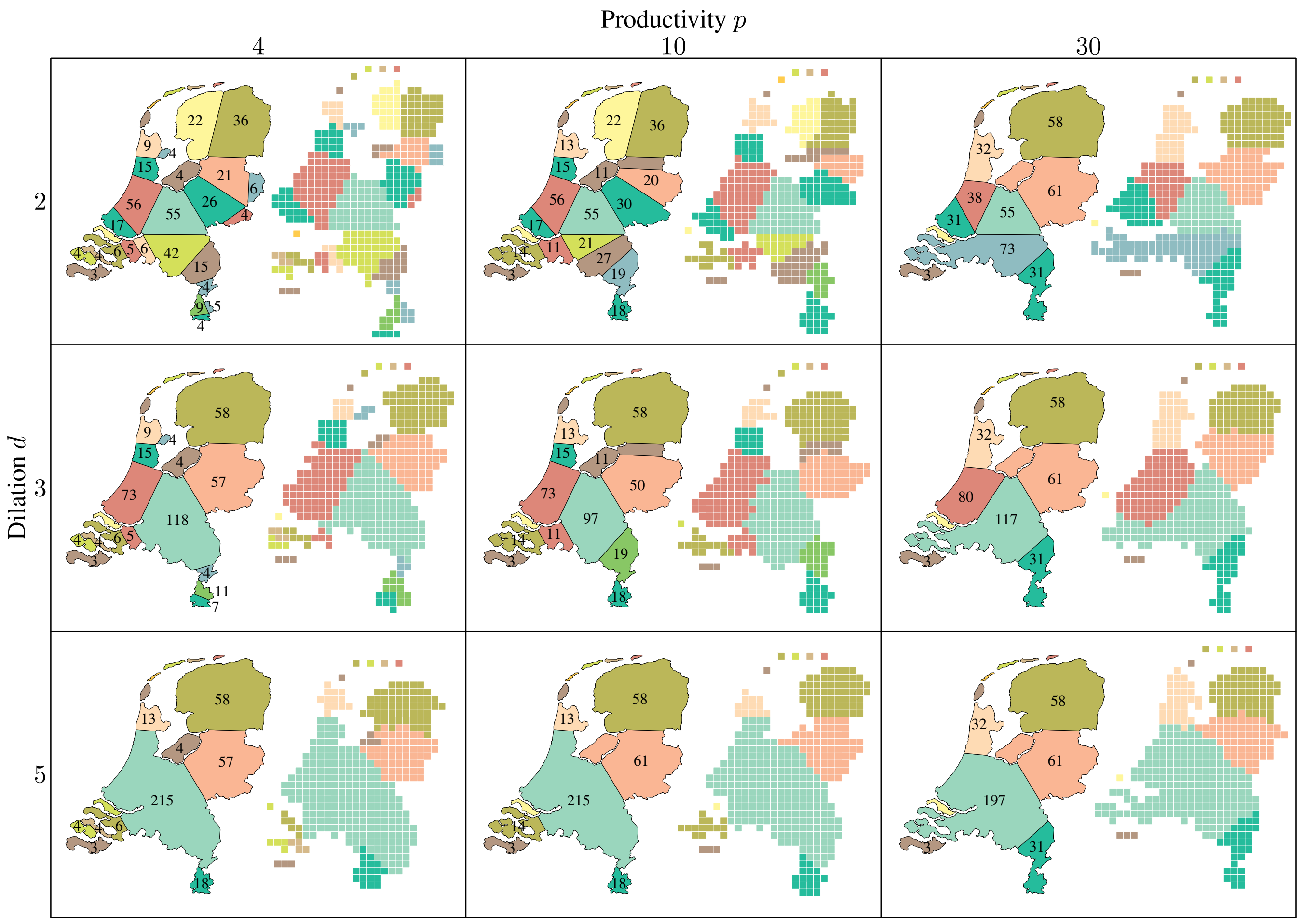

Fig. 12. Results of varying productivity and dilation thresholds on the decomposition and subsequent Mosaic Cartogram.

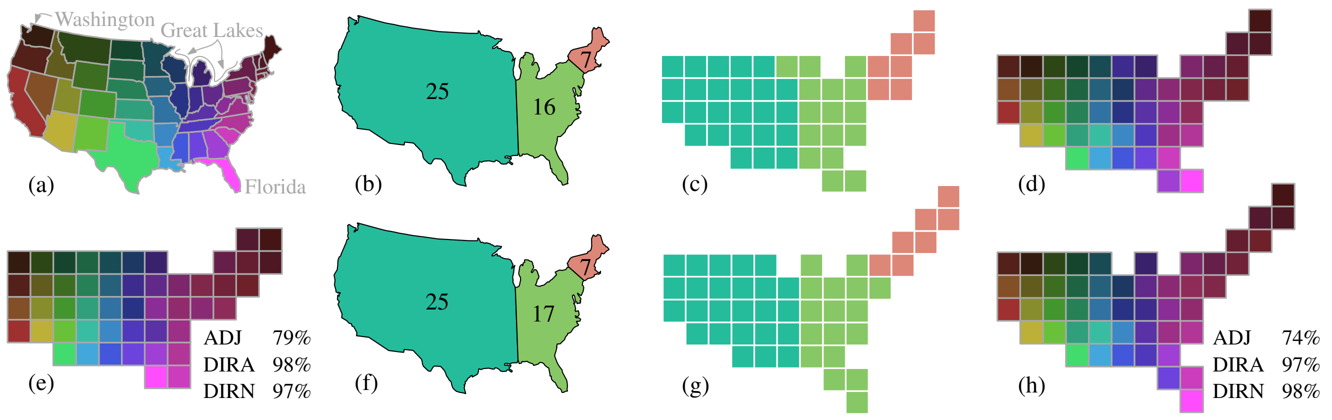

Fig. 13. (a) Contiguous states of the US. (b-d) Result computed by our pipeline ($d = 5, p = 4$). (e) Grid map based on Fig. 5a in [23], which includes Washington DC. (f-h) Result computed by our pipeline ($d = 5, p = 4$), for input including Washington DC.

which the metrics again further support, but our map has some points that we consider to work specifically well in comparison: Florida is the bottom-rightmost state; Michigan can be seen to be located between the Great Lakes. McNeill and Hale's result on the other hand nicely preserves Washington as the northwestern-most state, and is slightly more compact for the northeastern states.

Comparing Fig. 13(d,h), the addition of the single tile for Washington DC has quite an effect on the Mosaic Cartogram and thus the eventual result. Though the overall structure is the same, the shape around the Great Lakes is less clear in (d), but the northeastern states are represented more compactly. It is also worth noting that the projection of the input (here, Albers projection) for such large areas may have an effect on the coherence of the grid map [24].

Our method is somewhat slower to compute. For the 374 UK local authorities, McNeill and Hale report a running time of four seconds [23] for the result in Fig. 6. Our result shown in Fig. 7 takes 84 seconds to compute, with 21 seconds spent on generating the partition, and 60 seconds spent on computing a Mosaic Cartogram on a standard laptop. Improving this running time substantially thus requires a faster algorithm to compute Mosaic Cartograms.

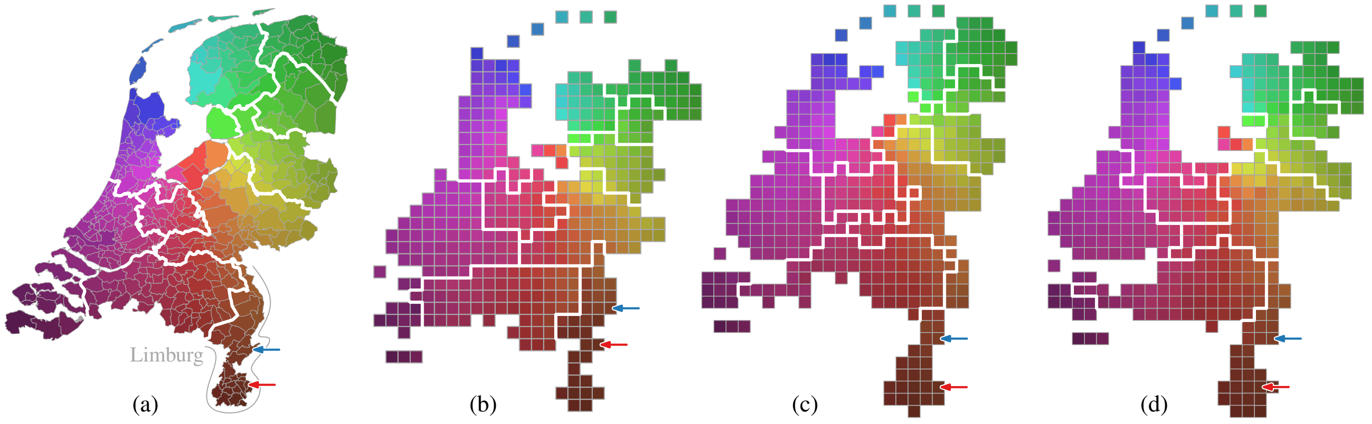

Fig. 14. Effect of applying a subdivision, illustrated with the Dutch municipalities and provinces. Thick white boundaries separate provinces. (a) Input map. (b) Result of our pipeline without decomposition, using provinces as parts. (c) Result of our pipeline ($d = 3, p = 4$) without using provinces. (d) Result of our pipeline ($d = 3, p = 10$) by applying decomposition to each province.

Subdivisions. So far, we have shown inputs of a single containing shape with spatial elements. However, the containing shape can often be subdivided into several administrative units: e.g., a country is often subdivided into provinces or states. The question is whether such known subdivisions of the containing shape can be used instead of, or in addition to, decomposition in our pipeline. To explore this, we consider again the Dutch municipalities. As these are hierarchically organized into provinces, we use province boundaries to subdivide the containing shape; see Fig. 14(a).

Fig. 14(b) shows the result of applying our pipeline without applying Step 1 – shape decomposition. Instead, we simply use the provinces as parts and start from Step 2. To compare, Fig. 14(c) shows our pipeline solution without subdividing by provinces, but with the province borders indicated. The main advantages of using the provinces as parts, is that (contiguous parts of) provinces remain contiguous in the result, and that adjacencies between provinces are exactly the same to those in the input map, as achieved by the Mosaic Cartogram. Without using the province subdivision, this is not guaranteed, and indeed we can observe slight discontiguities in Fig. 14(c) for some provinces. Note, however, that the provinces are only discontiguous when considering rook's adjacency (sharing a side of the square), but are in fact contiguous in queen's adjacency (sharing a corner is sufficient to be considered connected). As such, the deviation is indeed only minor.

At a glance, using the subdivision improves global shape in some places, such as southern Limburg (the southernmost province). However, this does not readily imply a good grid map: without decomposition, we see the same problems as discussed in Section 2.2 now appearing at a smaller scale: the tile indicated by the red arrow in Fig. 14(b) is assigned to the municipality in Fig. 14(a) indicated by the arrow of the same color; based on shape one would rather expect this tile to represent the municipality indicated by the blue arrow. That is, this map has poor coherence in terms of local shape. With decomposition, this improves significantly, as illustrated by the colored arrows for these same municipalities.

This raises the question whether we can effectively combine these two ideas. That is, rather than applying the decomposition step to the entire containing shape, we refine the subdivision into parts, by applying decomposition to each polygon it defines, i.e., to each province in our example. We then obtain parts that respect the subdivision; the Mosaic Cartogram ensures the correct topology between parts and thus we achieve contiguity for each province. The result of this is shown in Fig. 14(d). Whereas it indeed maintains contiguity of each province (and the entire shape), we see global shape deteriorating slightly, again mostly in terms of smoothness along the boundary. We attribute this to the increased number of parts with typically fewer spatial elements: as a result, the algorithm for Mosaic Cartograms is less sensitive to small changes along the boundary in capturing shape.

Future work. One major benefit of our pipelined approach is that improvements within each step can be carried over to improve grid-map computation – though especially for Step 1 it remains to be assessed how improvements in shape decomposition interact with the subsequent steps. We specifically see potential to improve Step 2. Whereas Mosaic Cartograms generally produce good tile arrangements, there are some limitations to the method. Specifically, the outline of the tile arrangement can capture shape well, but may also cause a somewhat jagged boundary where the containing shape did not necessarily feature such. As also observed in [23], boundary smoothness can improve grid-map quality. We expect that an extra post-processing step of the Mosaic Cartogram may reduce the jagged boundary.

Our pipeline also makes it straightforward to interact with the intermediate results: one can manually add or remove cuts in Step 1 that do not quite follow readily from the input geometry, but rather from a user's understanding of the local geography – as it is well known that small geometric features on a map may be relevant from a cultural or geographic point of view. Especially the Mosaic Cartogram can be interacted with easily and in a predictable manner. By simply shifting and swapping tiles, one can steer the input for the final step to an even more polished result. With Step 3 being efficiently computed, this can be done interactively with the tile assignment being updated on-the-fly, to ensure the best result. We emphasize that all results in this paper were not changed manually, beyond improving the placement of separate pieces (islands) of the Mosaic Cartogram (see Section 5).

We have compared our results to that of McNeill and Hale [23] qualitatively. A general and impartial quantitative evaluation would be useful, either in the form of user studies or computational experiments. The difficulty in achieving the latter is how to capture each facet of coherence well. Though Meulemans et al. [24] provide a suite of metrics, global shape poses a particular challenge: common shape similarity measures are not suitable for handling distortion which is a necessity in obtaining a high-quality grid map. Yet, not all distortion is equal; it must be applied effectively and in a structured way as to still have a sense of local scale for salient features, even though the larger structures are distorted. The ideal of "no distortion" in shape is not only unachievable but also undesirable. That is, current measures cannot easily operate within the context of density differences that force the distortion. Possibly, ideas from focus-and-context maps to measure distortion based on a local scale [16,47] might be useful in designing a locally scale-aware shape-similarity measure. Local shape similarly poses challenges, as it relies on automatically identifying and relating characteristic features in both the grid map and the input map.

ACKNOWLEDGMENTS

M. Sondag and B. Speckmann are (partially) supported by the Dutch Research Council (NWO) under project no. 639.023.20.

REFERENCES

[1] B. Berkowitz and L. Gamio. What you need to know about the measles outbreak. https://www.washingtonpost.com/graphics/health/how-fast-does-measles-spread/, February 2015. Accessed April 2020.

[2] P. Blickle and S. Venohr. Dürfen wir vorstellen: Deutschlands Muslime. http://www.zeit.de/gesellschaft/2015-01/islam-muslime-in-deutschland, January 2015. Accessed April 2020.

[3] H. Blum. A transformation for extracting new descriptors of shape. *Models for the perception of speech and visual form*, 19(5):362–380, 1967.

[4] Q. W. Bouts, I. Kostitsyna, M. J. van Kreveld, W. Meulemans, W. Sonke, and K. Verbeek. Mapping polygons to the grid with small Hausdorff and Fréchet distance. In *Proc. 24th Annual European Symposium on Algorithms*, vol. 57, pp. 22:1–22:16, 2016. doi: 10.4230/LIPIcs.ESA.2016.22

[5] F.-J. Brandenburg. On the complexity of optimal drawings of graphs. In *Proc. Graph-Theoretic Concepts in Computer Science*, vol. 411, pp. 166–180, 1989.

[6] R. G. Cano, K. Buchin, T. Castermans, A. Pieterse, W. Sonke, and B. Speckmann. Mosaic drawings and cartograms. *Computer Graphics Forum*, 34(3):361–370, 2015. doi: 10.1111/cgf.12648

[7] B. Casselman and A. McCann. Where your state gets its money. http://fivethirtyeight.com/features/where-your-state-gets-its-money/, April 2015. Accessed April 2020.

[8] F. Chin, J. Snoeyink, and C. A. Wang. Finding the medial axis of a simple polygon in linear time. *Discrete & Computational Geometry*, 21(3):405–420, 1999.

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

[10] D. DeBelius. Let's tesselate: Hexagons for tile grid maps. http://blog.apps.npr.org/2015/05/11/hex-tile-maps.html, May 2015. Accessed April 2020.

[11] D. Eppstein, M. J. van Kreveld, B. Speckmann, and F. Staals. Improved grid map layout by point set matching. *International Journal on Computational Geometry Applications*, 25(2):101–122, 2015. doi: 10.1142/S0218195915500077

[12] B. Fong. How to make tile grid maps in Tableau. https://public.tableau.com/s/blog/2016/01/how-make-tile-grid-maps-tableau, January 2016. Accessed April 2020.

[13] M. T. Gastner and M. E. J. Newman. Diffusion-based method for producing density-equalizing maps. *Proc. National Academy of Sciences*, 101(20):7499–7504, 2004. doi: 10.1073/pnas.0400280101

[14] M. T. Gastner, V. Seguy, and P. More. Fast flow-based algorithm for creating density-equalizing map projections. *Proc. National Academy of Sciences*, 115(10):E2156–E2164, 2018. doi: 10.1073/pnas.1712674115

[15] D. Guo, J. Chen, A. M. MacEachren, and K. Liao. A visualization system for space-time and multivariate patterns (vis-stamp). *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1461–1474, 2006.

[16] J. Haunert and L. Sering. Drawing road networks with focus regions. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2555–2562, 2011. doi: 10.1109/TVCG.2011.191

[17] D. D. Hoffman and W. Richards. Parts of recognition. *Cognition*, 18(1):65–96, 1984.

[18] L. J. Latecki and R. Lakämper. Convexity rule for shape decomposition based on discrete contour evolution. *Computer Vision and Image Understanding*, 73(3):441–454, 1999.

[19] X. Liu, Y. Hu, S. North, and H. Shen. CorrelatedMultiples: Spatially coherent small multiples with constrained multi-dimensional scaling. *Computer Graphics Forum*, 37(1):7–18, 2018.

[20] M. Löffler and W. Meulemans. Discretized approaches to schematization. In *Proc. 29th Canadian Conference on Computational Geometry*, pp. 220–225, 2017.

[21] Y. Lu, J. Lien, M. Ghosh, and N. M. Amato. α-decomposition of polygons. *Computers & Graphics*, 36(5):466–476, 2012.

[22] B. B. Mandelbrot. *How Long Is the Coast of Britain*, pp. 25–33. W. H. Freeman, New York, 1983.

[23] G. McNeill and S. A. Hale. Generating tile maps. *Computer Graphics Forum*, 36(3):435–445, 2017.

[24] W. Meulemans, J. Dykes, A. Slingsby, C. Turkay, and J. Wood. Small multiples with gaps. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):381–390, 2017. doi: 10.1109/TVCG.2016.2598542

[25] X. Mi and D. DeCarlo. Separating parts from 2d shapes using relatability. In *IEEE International Conference on Computer Vision*, pp. 1–8, 2007.

[26] New York Times. How the rulings affect gay couples. http://www.nytimes.com/interactive/2013/06/26/us/scotus-gay-marriage.html, June 2013. Accessed April 2020.

[27] N. Papanelopoulos, Y. Avrithis, and S. Kollias. Revisiting the medial axis for planar shape decomposition. *Computer Vision and Image Understanding*, 179:66–78, 2019.

[28] H. Park. Gay marriage state by state: From a few states to the whole nation. http://www.nytimes.com/interactive/2015/03/04/us/gay-marriage-state-by-state.html, March 2015. Accessed April 2020.

[29] K. Powell, R. Harris, and F. Cage. How voter-friendly is your state? http://www.theguardian.com/us-news/ng-interactive/2014/oct/22/-sp-voting-rights-identification-how-friendly-is-your-state, October 2014. Accessed April 2020.

[30] R. Radburn. Go with the flow: Commuting & migration flows within London. https://public.tableau.com/profile/robradburn/#!/vizhome/ODMpasLondonAftertheFlood/GowiththeFlow, March 2016. Accessed April 2020.

[31] R. Radburn. Home truths in London. https://public.tableau.com/profile/robradburn/#!/vizhome/HomeTruthsinLondon/LondonTenure, April 2020. Accessed April 2020.

[32] N. Richards. Where are the Africa grid/tile maps? https://questionsindataviz.com/2016/09/01/where-are-the-africa-gridtile-maps/, September 2016. Accessed April 2020.

[33] N. Richards. How do you tile the world? https://questionsindataviz.com/2017/11/05/how-do-you-tile-the-world/, November 2017. Accessed April 2020.

[34] N. Richards. Do tile maps need to have regular shapes? https://questionsindataviz.com/2019/02/02/do-tile-maps-need-to-have-regular-shapes/, February 2019. Accessed April 2020.

[35] N. Richards. When are two maps better than one? https://questionsindataviz.com/2019/05/21/when-are-two-maps-better-than-one/, May 2019. Accessed April 2020.

[36] T. Shaw. Good data visualization practice: Tile grid maps. https://www.forumone.com/ideas/good-data-visualization-practice-tile-grid-maps-0/, April 2016. Accessed April 2020.

[37] K. Siddiqi and B. B. Kimia. Parts of visual form: Computational aspects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):239–251, 1995.

[38] M. Singh and D. D. Hoffman. Part-based representations of visual shape and implications for visual cognition. *Advances in psychology*, 130:401–459, 2001.

[39] M. Singh, G. D. Seyranian, and D. D. Hoffman. Parsing silhouettes: The short-cut rule. *Perception & Psychophysics*, 61(4):636–660, 1999.

[40] A. Slingsby. Tilemaps for summarising multivariate geographical variation. In *Proc. Workshop on Visual Summarization and Report Generation*, 2018.

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

[42] A. Slingsby, M. Kelly, and J. Dykes. OD maps for showing changes in Irish female migration between 1851 and 1911. *Environment and Planning A*, 46(12):2795–2797, 2014. doi: 10.1068/a140112g

[43] A. Slingsby, J. Wood, and J. Dykes. Treemap cartography for showing spatial and temporal traffic patterns. *Journal of Maps*, 6(1):135–146, 2010.

[44] A. Tribou and K. Collins. This is how fast America changes its mind. http://www.bloomberg.com/graphics/2015-pace-of-social-change/, June 2015. Accessed April 2020.

[45] E. R. Tufte. *The Visual Display of Quantitative Information*, vol. 2. Graphics Press Cheshire, CT, 1983.

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

[47] T. C. van Dijk and J. Haunert. Interactive focus maps using least-squares optimization. *International Journal of Geographical Information Science*, 28(10):2052–2075, 2014. doi: 10.1080/13658816.2014.887718

[48] K. Wongsuphasawat. A semi-automatic way to create your own grid map. https://medium.com/kristw/creating-grid-map-for-thailand-397b53a4ecf, January 2016. Accessed April 2020.

[49] K. Wongsuphasawat. Whose grid map is better? quality metrics for grid map layouts. https://medium.com/@kristw/whose-grid-map-is-better-quality-metrics-for-grid-map-layouts-e3d6075d9e80, January 2016. Accessed April 2020.

[50] J. Wood, D. Badawood, J. Dykes, and A. Slingsby. BallotMaps: Detecting name bias in alphabetically ordered ballot papers. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2384–2391, 2011.

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

[52] J. Wood, A. Slingsby, and J. Dykes. Visualizing the dynamics of London's bicycle-hire scheme. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 46(4):239–251, 2011.