Metaphoric maps for dynamic vertex-weighted graphs

T. Mchedlidze [1] and C. Schnorr [2]

[1]Utrecht University, Department of Information and Computing Sciences, Netherlands
[2] Apple, Boulder, USA

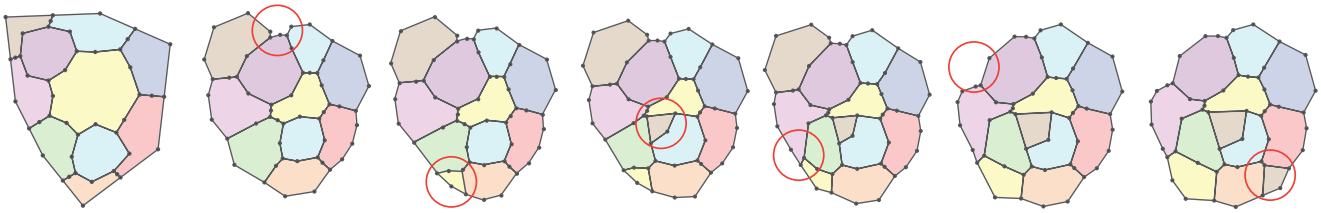

Figure 1: *Dynamic map for a random input and random sequence of operations: edge–, vertex+, vertex+, edge+, vertex–, vertex+*

Abstract

In this paper we study metaphoric maps of dynamic vertex-weighted graphs. Dynamic operations on such graphs allow a vertex to change the weight, vertices and edges appear and disappear. In the metaphoric map this is viewed as country shrink and growth, appearance and disappearance and change in the country adjacency. We present a force-based algorithm that supports these operations. In the design of the algorithm we prioritize the dynamic stability of the map, the accuracy in the size of countries and low complexity of the polygons representing the countries. We evaluate the algorithm based on the state-of-the-art quality metrics for randomly generated inputs of various complexity.

1. Introduction

Visualization of dynamic graphs is a prominent topic in information visualization. Multiple algorithms have been designed for visualization of dynamic graphs in form of node-link and matrix representations [BBDW17]. Map-like graph visualizations, where vertices are represented by polygonal regions and edges by contacts among those, have also gained certain popularity [HHS20], by making advantage of the human familiarity with geographic maps. These visualisations have been seen to outperform treemaps in the tasks that require recognition of hierarchy [BPP17] and found more enjoyable than node-link visualizations [SSK16]. An important advantage of these visualizations is that they can easily display the weights associated with the vertices, by the mean of resizing the map's regions. Such visualizations are also known as *area-proportional contact representations* [Ala15] and are closely relevant to cartograms [Tob04,NK16]. By Hogräfer et al. [HHS20], map-like representations of graphs and cartograms lie on two opposite sides of a single spectrum of map-like visual representations. Unlike the dynamic visualization of graphs [BBDW17] and dynamic cartograms [KNP04, CM08], dynamic map-like visualizations of graphs have seen little attention. In particular, while the operation of weight change has been well-studies in the cartogram literature, other dynamic changes, such as addition/removal of vertices and edges, have not been unified under a single framework.

In this paper, we address this gap in the literature and present an algorithm based on a force-directed simulation that produces an animated map-like visualization of a dynamic vertex-weighted graph. To make the task of comparing the areas of regions possible, we target to obtain regions that have relatively simple shapes. To ensure the dynamic stability, we require that the dynamic graph differs between the two time stamps as little as possible, in particular, by a single *dynamic operation*. We define the set of basic operations on vertex-weighted dynamic graphs that are meaningful for maps-like visualization (Section 3). Our algorithm works in two phases: the *static phase* (Section 2), that produces a statistically accurate map and the *dynamic phase* (Section 3), that implements the dynamic operations. We evaluate our algorithm (Section 4) based on a measure of polygon complexity [BKSB95] and on the state-of-the-art measure of statistical accuracy of cartograms [AKV15]. The code of our implementation is publicly available [Sch20].

There exist theoretical approaches on achieving map-like visualizations with absolute statistical accuracy [Tho92,ABF*13,Kle18], which however are not guaranteed from regions with narrow corridors that contradicts our goal of simply looking regions. Dynamic cartograms [KNP04, CM08] allow only the operation of weight change, and insist on preserving geographic accuracy, that is not relevant for map-like representations of graphs. GMap [GHK10] constructs map-like visualization of clustered graphs that had been

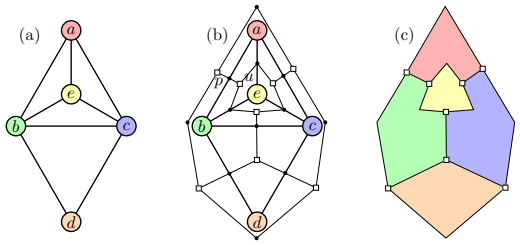

Figure 2: *(a) A planar drawing of a graph G_0. (b) Placement of dual vertices and bend-vertices. (c) The resulting map.*

extended to work in dynamic setting [HKV14]. Here the dynamic changes concern the clustered graph as opposed to our setting where the dynamic changes are applied to the graph expressing the adjacencies of the map. Additionally GMap does not try to achieve accurate cartographic accuracy explicitly or to control the shapes of the clusters. Simonetto et al's force-directed algorithm [SAS16] uses forces partially similar to ours to construct Euler diagrams with smooth region boundaries. The algorithm does not target to achieve simply looking regions or to resize them, neither does it address the dynamic setting.

A dynamic graph is defined as a sequence $\Gamma = (G_0, G_2, ..., G_{n-1})$ where $G_i := (V_i, E_i, w_i)$ is a vertex-weighted graph, $w_i : V_i \to \mathbb{R}^+$ it's weight function, and indices refer to a sequence of time steps. In a *metaphoric map* M_i of G_i vertices are depicted by polygons (*regions*), the adjacent vertices appear as polygons sharing a non-trivial boundary, finally each polygon's area roughly corresponds to the corresponding vertex's weight. *Statistical accuracy* of M_i describes how closely the areas of its regions match their weights. Graph G_i is referred to as *contact graph* of M_i. Throughout the paper we treat a metaphoric map as a planar graph, the vertices and the edges of which are represented by the union of the vertices and edges of map's polygons.

2. Generation of the map

In this section we describe the static phase of the algorithm, that given a vertex-weighted graph, constructs its metaphoric map with the goal to achieve high statistical accuracy and, at the same time, to make the regions to have low complexity.

Initial drawing of the map This first step of the static phase takes as input a planar graph G_0 with a planar drawing Γ and produces a map M of G_0. We rely on the notion of the dual graph of G_0 to produce the map M. However, we treat the outer face of G_0 in a special way, by placing there as many vertices as the edges on the outer face of G_0 (instead of one vertex, as dictated by the definition of the dual). Fig. 2 illustrates this procedure.

Force-directed steps Let M be the map produced in the first step, F be the set of internal regions of M, $w_0 : F \to \mathbb{R}_+$ be the weight function. In the second step, we apply a force-directed graph drawing algorithm to M to reduce the *cartographic error* $|A(g) - w_0(g)|$ for every region $g \in F$ by thus producing map M_0. Here $A(g)$ denotes the area of region g. It is a hard requirement for this step to preserve the planar embedding of M. As explained above, we also want the regions of M_0 to be visually simple. Our intuitive understanding of polygon visual simplicity is that a region has a round-looking shape

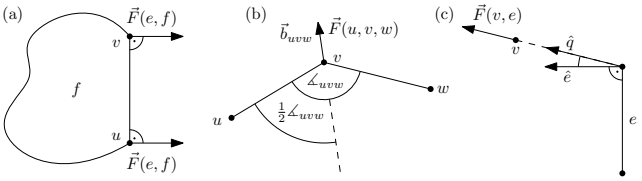

Figure 3: *(a) Forces explained by the air-pressure. (b) Angular resolution force. (c) Vertex-edge repulsion force.*

and thus does not have narrow corridors. For the formalization of this notion refer to Section 4. Both the reduction of cartographic error and the achievement of visual simplicity are soft requirements. We now describe the concrete force components. The constant factors of the forces were determined experimentally.

Air pressure Similarly to Alam et al. [ABF*13], we treat the polygonal regions as volumes of some amount of air equal to the respective region's weight. To eliminate the influence of the constant factors of the region weights, we normalize the pressure $P(g)$ in an internal region g as $P(g) = \frac{w_0(g)}{A(g)} \frac{\sum_{f\in F} A(f)}{\sum_{f\in F} w_0(f)}$. We set the pressure in the outer region g_o to $P(g_o) = \frac{\sum_{f\in F} A(f)P(f)}{\sum_{f\in F} A(f)} = 1$. The air pressure in region g exerts a force on each bounding edge e based on the pressure's magnitude and the edge length $\ell(e)$ in relation to the length of the entire region boundary $\ell(g)$. Therefore the force on edge e from region g is $\vec{F}(e,g) = 3P(g)\frac{\ell(e)}{\ell(g)}\hat{r}$, where $\hat{r}$ is a unit vector perpendicular to e directed towards outside of g, Fig. 3(a).

Angular resolution Internal region angles close to 0° and 360° cause narrow corridors. Thus, inspired by Argyriou et al. [ABS13], we define a force that tries to evenly distribute the angles around a vertex. For a vertex v, we define the force $\vec{F}(u,v,w) = \frac{1}{2}\frac{\frac{360^\circ}{\deg(v)} - \alpha_{uvw}}{\alpha_{uvw}}\hat{b}_{uvw}$, where α_{uvw} denotes the angle $\angle uvw$ and $\hat{b}_{uvw}$ is the normalized bisector of $\angle wvu$, refer to Fig. 3(b).

Vertex-vertex repulsion Similarly to Eades [Ead84], we define a repulsive force between pairs of vertices to prevent the vertices from clumping together $\vec{F}(u,v) = 25\frac{1}{\|\vec{uv}\|^2}\widehat{uv}$. Differently from [Ead84], we apply this force to adjacent vertices too, because we do not try to achieve a uniform edge length.

Vertex-edge repulsion In another attempt to prevent narrow corridors, we define a repulsive force between vertices and edges. Given an edge e and non-incident vertex v, let $\vec{q}$ denote the vector connecting v to the point on e with the smallest Euclidean distance to v, and $\widehat{e}$ denote the unit vector perpendicular to edge e. We define this force as $\vec{F}(v,e) = 10\frac{1}{\|\vec{q}\|^2}(\widehat{e}\cdot\widehat{q})\widehat{q}$, refer to Fig. 3(c). Because of the component $\widehat{e}\cdot\widehat{q}$ the force deteriorates as the angle between $\widehat{e}$ and $\widehat{q}$ tends to 90°. This force differs from the one defined by Bertault [Ber00] in that it has an additional dot product term that measures how v is positioned relative to the edge e. In our experiments this led to more stable performance.

Both repulsion forces are applied on the elements on the same face. Due to preserved planarity of the map this is enough to push elements and improves the performance. Let $\bar{\ell}$ denote the average edge length. We eliminate vertices of degree 2 that become closer than $\frac{1}{10}\bar{\ell}$ to their neighbor, if we can do so without introducing edge crossings. We split in half the edges that get longer than $2\bar{\ell}$, in

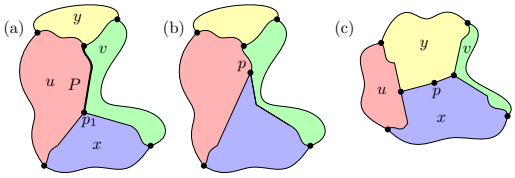

Figure 4: *Edge flip.*

order to create more degrees of freedom for the region shapes. In deciding how much a vertex can be displaced, we adopt the rules of ImPrEd [SAAB11] that ensure that no vertex crosses over an edge, which leads to the preservation of the planar embedding of the map.

3. Dynamic changes

We restrict the algorithm to the maps where: 1. no more than three regions meet at a point; 2. the boundary of the map is a simple polygon; 3. the common boundary of two regions is contiguous. These are practically reasonable restrictions, which allow us to slightly limit the scope of our experiments. These restrictions imply that G is a simple, planar, biconnected and internally triangulated graph. Thus, we only consider *basic dynamic operations* that preserve these properties, i.e.: change of the vertex weight, insertion/removal of an edge in the outer face, edge flip, vertex insertion/removal. Known that any two triangulations with k vertices can be transformed to each other by means of at most $O(k)$ edge flips [CHK*18], it is possible to represent more complex operations as sequences of the given basic operations. Basis dynamic operations are integrated into the force-directed simulations as follows. We pause the simulation, perform an operation on the current map M_i, producing map M_{i+1} and then resume the simulation on M_{i+1}.

Edge flip Consider an internal edge (u,v) of G_i that is incident to two internal faces f and g. Let x and y be the third vertices bounding f and g, resp. Since G_i is simple, $x \neq y$. The operation of *edge flip* replaces the edge (u,v) by the edge (x,y). To preserve the simplicity of G_i, flipping (u,v) is permitted only if $(x,y) \notin G_i$. An edge flip translates to region adjacency being flipped in the map. An edge flip is performed in map M_i in two phases; Fig. 4. First, we contract the boundary P between regions u and v into a single point p. Then, we expand p into a segment representing a boundary between x and y. During this procedure we make sure that no new crossings are introduced by means of adding bend-vertices.

Inserting and removing edges on the outer face The edge flip operation describes how internal regions change adjacencies. However, the same should be possible for external regions. This corresponds to the operation of edge insertion and removal in the outer face. In order to preserve G_i being internally triangulated, inserting (u,w) is only permitted if there exists a vertex v, and (u,v) and (v,w) which lie on the outer face. Removing an edge (u,w) on the outer face of the contact graph is only permitted if the graph remains biconnected, which happens if the third vertex v in the internal face bounded by (u,w) does not lie on the outer face. Both of these operations are a special case of the edge flip operation.

Inserting a vertex We discuss the case of inserting into internal face, with external face being a special case. All internal faces of the contact graph G_i are triangles. If we add a vertex x into face

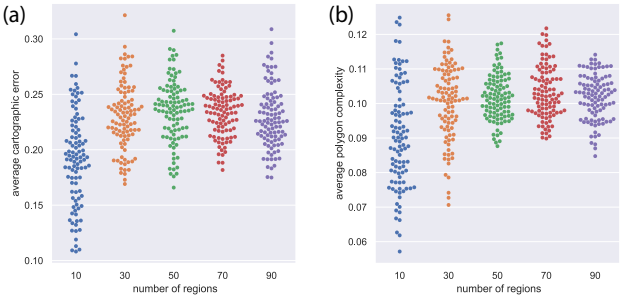

Figure 5: *(a) Cartographic error and (b) polygon complexity for 100 random instances of varying size after 200 force-directed steps.*

bounded by u, v, and w, we also add edges (x,u), (x,v), (x,w), to preserve the triangulatedness. Let p_{uvw} be the vertex of the map M_i that is common to regions u, v, and w in M_i; Also, let p_{uv}, p_{vw}, and p_{wu} denote the bend-vertices on the boundary of these faces that are incident to p_{uwv}. If any of the boundaries consist of only one edge, we subdivide it at its midpoint first. We then remove the vertex p_{uvw} along with its incident edges and insert edges between the bend-vertices p_{uv}, p_{vw}, p_{wu} instead, in order to bound a new region x. To avoid crossings we add bends-vertices to those edges.

Removing vertex When removing an internal vertex x from G_i, we check whether $\deg(x) = 3$ i.e. G_{i+1} stays internally triangulated. Let u, v, and w be the three neighbors of x. In order to avoid a hole in the map M_{i+1}, the three regions u, v, and w must take over the area of region x. Thus we remove the boundary between x and one of u, v and w. In practice, the removal operation is barely noticeable, because region x gets smaller before it disappears from the map. Removing a vertex on the outer face of G_i is a special case of the above but is performed only when G_{i+1} stays biconnected.

4. Evaluation

Quality metrics The design and evaluation of our algorithm was driven by the goals of ensuring the dynamic stability of the algorithm, the statistical accuracy of the maps and the reduction of the visual complexity of the regions. The dynamic stability is ensured by the design – the algorithm is based on the force-directed simulation and the operations perform minimal changes to the map; Fig.1.

Let $w(v)$ and $A(v)$ denote the desired and the actual area of the region v. To evaluate the statistical accuracy of the map, we use metric *normalized cartographic error* [AKV15]: $\max_{v \in V(G)} \frac{|A'(v)-w(v)|}{\max\{A'(v),w(v)\}}$, here $A'(v) := A(v) \cdot \frac{\sum_{u \in V} w(u)}{\sum_{u \in V} A(u)}$.

To quantify the complexity of regions, we rely on the measure proposed by Brinkhoff et al. [BKSB95], composed of three ingredients which we define next. Let P be a polygon with n vertices and let hull(P), $A(P)$ and circ(P) denote the convex hull, the area and the length of the boundary, resp. Let $L(P)$ be the number of P's internal angles that are greater than 180°, and let $L'(P) := L(P)/n - 3$ with $L'(P) := 0$ in case that $n \leq 2$. The *frequency of the vibration* of P is defined as $\text{freq}(P) := 1 + 16 \cdot (L'(P) - 0.5)^4 - 8 \cdot (L'(P) - 0.5)^2 \in [0,1]$ and its *amplitude* as $\text{ampl}(P) := \frac{\text{circ}(P) - \text{circ}(\text{hull}(P))}{\text{circ}(P)} \in [0,1]$. Function freq($P$) is 0 for

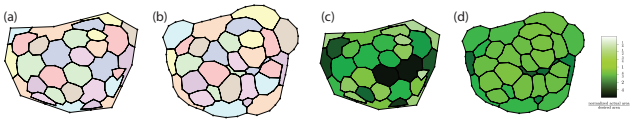

Figure 6: *(a) A map after generation, (b) after 200 of force-directed steps, (c-d) the air pressure in the regions before and after.*

a convex polygon and is maximum when half of the angles are large. Function $\mathrm{ampl}(P)$ is also 0 for a convex polygon and tends to 1 when the boundary of the polygon is getting longer comparably to the length of the convex hull. To measure the *convexity of a polygon*, we slightly modify the measure proposed by Brinkhoff et al. in order to differentiate between lengthy and fat convex polygons. Thus we measure the fraction of area the polygon P does not cover in its smallest enclosing circle, denoted by $\mathrm{encirc}(P)$. To keep the metric in the unit interval, we actually compare P's area to the maximal area of a regular n-gon in said smallest enclosing circle: $\mathrm{conv}(P) := 1 - \frac{A(P)}{A(\mathrm{encirc}(P)) \cdot \sin\left(\frac{360°}{n}\right) \cdot \frac{n}{2\pi}} \in [0,1]$ Brinkhoff et al. combine these three properties as $\mathrm{compl}(P) := 0.8 \cdot \mathrm{ampl}(P) \cdot \mathrm{freq}(P) + 0.2 \cdot \mathrm{conv}(P) \in [0,1]$ According to them, measure $\mathrm{compl}(P)$ is able to capture the intuitive perception of a polygon's complexity [BKSB95]. In our tests, this slightly modified measure, which we refer to as *polygon complexity*, has shown to align with our intuitive understanding of visual complexity.

Test data In order to evaluate our algorithm, we have generated a set of planar contact graphs, each accompanied with a planar straight-line drawing thereof and a sequence of dynamic operations to be applied to it. To see how the algorithm performs for the inputs of various complexity we explored the following observation: if the contact graph of a map contains a cycle of length k with a vertices in this cycle, the map will need to contain a ring composed of k polygons with a polygons inside it. The smaller the k and the larger the a, the harder is to create a map without thin regions. To implement this idea, we construct a Delaunay triangulation of random point sets and iterate on this process in some of the triangles. The parameter *nesting ratio* $\alpha \in [0,1]$ determines what fraction of the vertices are nested into other triangles; The parameter *nesting bias* $\beta \in [0,1)$ determines the depth of the nesting.

Results of the experiments In our experiments we were driven by three questions: 1. What is the quality of the maps after the static phase? 2. Does the quality of the maps depend on the complexity of the contact graphs, when the complexity is measured by graph size, nesting ratio and nesting bias? 3. Does the quality of the map changes when dynamic operations are applied? Applying our algorithm to graphs with up to 90 vertices, we have seen that the quality of the map stabilizes after 200 force-directed steps, that average cartographic error is below 0.3 and that neither the cartographic error nor polygon complexity are strongly affected by the graph size; Fig 5. It has been proven that maps we consider in this paper can be modified to have cartographic error zero [Tho92, ABF*13], however, in both cases regions can have long and skinny parts and no theoretical guarantees are known on the fatness of the regions. Thus, our algorithm is not achieving close to optimal cartographic error but this comes with the advantage of simply looking regions with low polygon complexity. A map with 30 regions after the generation and after 200 force-directed steps is illustrated in Fig. 6. We observe that most of the regions have a simply looking organic shape and the air-pressure significantly decreases. There are several regions that are lengthy, but they have small areas and complex pattern of adjacencies, which explains their more complex shape. The running time of the algorithm depends on the number of force-directed steps, thus if we fix those to 200 the time is mostly below 8 seconds and the quality of the map remains stable with the increase of the size; compare Fig. 5 and Fig. 7.

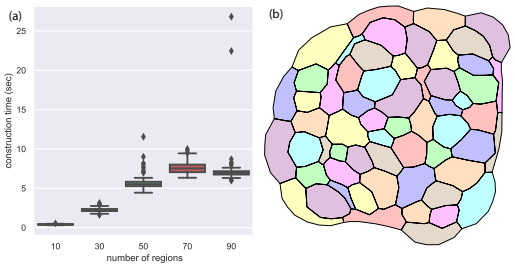

Figure 7: *(a) Construction time with 200 force-directed steps. (b) Map with 90 regions.*

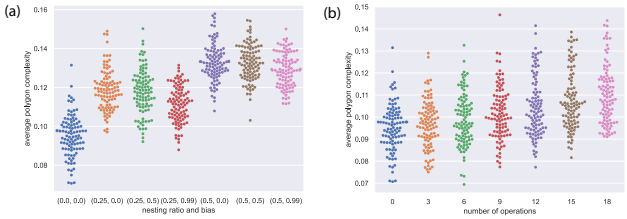

Figure 8: *Polygon complexity for 100 random instances with $n = 20$ (a) with different nesting ratios α and nesting biases β, (b) after applying a number of operations with $\alpha = \beta = 0$.*

We observed a pronounced impact of the nesting ratio α on the polygon complexity; Fig. 8.a. This confirms the observation that small cycles in contact graph with many vertices embedded in those cycles, force the regions of the cycle to be complex. When applying dynamic operations on the map, we have observed that the cartographic error initially decreases. We believe that this occurs partially due to the increased amount of vertices on the regions boundaries, which allow for more complex shapes and therefore better accuracy and partially due to the fact that force-directed simulation has more time to optimize the layout. The polygon complexity, on the other hand, increases over time; Fig. 8.b. Recall that the new vertices in the contact graph are added by nesting them into triangles, by leading to the increase in the nesting ratio, and thus explains the increase in the polygon complexity.

Conclusion The presented algorithm is targeting visualisation of dynamic vertex-weighted graphs. However, it can be also used to visualize dynamic clustered graphs, by using the cluster sizes as weights. In this view, it is intriguing to compare the performance of our algorithm to dynamic GMap [HKV14]. Another research direction is to study whether the polygon complexity metric is an appropriate measure for quantifying complexity of map-like visualizations and indeed correlates with the human perception in this setting.

References

[ABF*13] ALAM M. J., BIEDL T., FELSNER S., KAUFMANN M., KOBOUROV S. G., UECKERDT T.: Computing cartograms with optimal complexity. *Discrete & Computational Geometry 50*, 3 (2013), 784–810. doi:10.1007/s00454-013-9521-1. 1, 2, 4

[ABS13] ARGYRIOU E. N., BEKOS M. A., SYMVONIS A.: Maximizing the total resolution of graphs. *Comput. J. 56*, 7 (2013), 887–900. doi:10.1093/comjnl/bxs088. 2

[AKV15] ALAM M. J., KOBOUROV S. G., VEERAMONI S.: Quantitative measures for cartogram generation techniques. *Comput. Graph. Forum 34*, 3 (2015), 351–360. 1, 3

[Ala15] ALAM M. J.: *Contact representations of graphs in 2D and 3D*. Doctoral thesis, The University of Arizona, 2015. URL: https://www.proquest.com/openview/5bf2b2dc7f21bf9f8759bdc89632b50b/1?pq-origsite=gscholar&cbl=18750. 1

[BBDW17] BECK F., BURCH M., DIEHL S., WEISKOPF D.: A taxonomy and survey of dynamic graph visualization. *Comput. Graph. Forum 36*, 1 (2017), 133–159. doi:10.1111/cgf.12791. 1

[Ber00] BERTAULT F.: A force-directed algorithm that preserves edge-crossing properties. *Inf. Process. Lett. 74*, 1-2 (2000), 7–13. doi:10.1016/S0020-0190(00)00042-9. 2

[BKSB95] BRINKHOFF T., KRIEGEL H.-P., SCHNEIDER R., BRAUN A.: Measuring the complexity of polygonal objects. In *ACM-GIS* (1995), p. 109. 1, 3, 4

[BPP17] BIUK-AGHAI R. P., PANG P. C., PANG B.: Map-like visualisations vs. treemaps: an experimental comparison. In *Proceedings of the 10th International Symposium on Visual Information Communication and Interaction, VINCI 2017, Bangkok, Thailand, August 14-16, 2017* (2017), Biuk-Aghai R. P., Li J., Takahashi S., (Eds.), ACM, pp. 113–120. doi:10.1145/3105971.3105976. 1

[CHK*18] CARDINAL J., HOFFMANN M., KUSTERS V., TÓTH C. D., WETTSTEIN M.: Arc diagrams, flip distances, and hamiltonian triangulations. *Comput. Geom. 68* (2018), 206–225. doi:10.1016/j.comgeo.2017.06.001. 3

[CM08] CARROLL G., MOORE A.: *A Multi-scale Dynamic Map Using Cartograms to Reflect User Focus*. Springer Berlin Heidelberg, 2008, pp. 87–112. doi:10.1007/978-3-540-70970-1_5. 1

[Ead84] EADES P.: A heuristic for graph drawing. *Congressus Numerantium 42* (1984), 149–160. 2

[GHK10] GANSNER E. R., HU Y., KOBOUROV S. G.: GMap: Visualizing graphs and clusters as maps. In *IEEE Pacific Visualization Symposium PacificVis 2010, Taipei, Taiwan, March 2-5, 2010* (2010), IEEE Computer Society, pp. 201–208. doi:10.1109/PACIFICVIS.2010.5429590. 1

[HHS20] HOGRÄFER M., HEITZLER M., SCHULZ H.: The state of the art in map-like visualization. *Comput. Graph. Forum 39*, 3 (2020), 647–674. doi:10.1111/cgf.14031. 1

[HKV14] HU Y., KOBOUROV S. G., VEERAMONI S.: Embedding, clustering and coloring for dynamic maps. *J. Graph Algorithms Appl. 18*, 1 (2014), 77–109. doi:10.7155/jgaa.00315. 2, 4

[Kle18] KLEIST L.: Drawing planar graphs with prescribed face areas. *J. Comput. Geom. 9*, 1 (2018), 290–311. doi:10.20382/jocg.v9i1a9. 1

[KNP04] KEIM D. A., NORTH S. C., PANSE C.: Cartodraw: A fast algorithm for generating contiguous cartograms. *IEEE Trans. Vis. Comput. Graph. 10*, 1 (2004), 95–110. doi:10.1109/TVCG.2004.1260761. 1

[NK16] NUSRAT S., KOBOUROV S. G.: The state of the art in cartograms. *Comput. Graph. Forum 35*, 3 (2016), 619–642. doi:10.1111/cgf.12932. 1

[SAAB11] SIMONETTO P., ARCHAMBAULT D., AUBER D., BOURQUI R.: Impred: An improved force-directed algorithm that prevents nodes from crossing edges. *Comput. Graph. Forum 30*, 3 (2011), 1071–1080. doi:10.1111/j.1467-8659.2011.01956.x. 3

[SAS16] SIMONETTO P., ARCHAMBAULT D., SCHEIDEGGER C.: A simple approach for boundary improvement of euler diagrams. *IEEE Trans. Vis. Comput. Graph. 22*, 1 (2016), 678–687. doi:10.1109/TVCG.2015.2467992. 2

[Sch20] SCHNORR C.: *Visualizing Dynamic Clustered Data Using Area-proportional Maps*. Master's thesis, Karlsruhe Institute of Technology, 2020. URL: https://github.com/jenox/Master-Thesis.git. 1

[SSK16] SAKET B., SCHEIDEGGER C., KOBOUROV S. G.: Comparing node-link and node-link-group visualizations from an enjoyment perspective. *Comput. Graph. Forum 35*, 3 (2016), 41–50. doi:10.1111/cgf.12880. 1

[Tho92] THOMASSEN C.: Plane cubic graphs with prescribed face areas. *Comb. Probab. Comput. 1* (1992), 371–381. doi:10.1017/S0963548300000407. 1, 4

[Tob04] TOBLER W.: Thirty five years of computer cartograms. *An. of the Assoc. of American Geographers 94*, 1 (2004), 58–73. doi:10.1111/j.1467-8306.2004.09401004.x. 1