GeomLab 2020

September 16, 2020

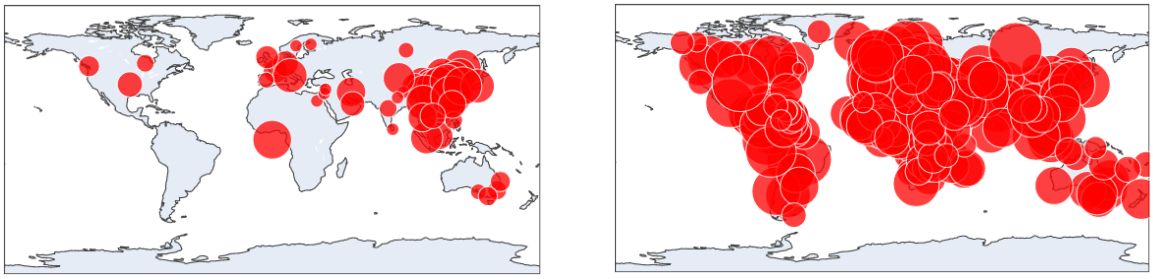
Contents

1 Motivation

In the wake of the COVID-19 pandemic, several outlets printed two-dimensional depictions of data with each symbol representing multiple pieces of information at once. These datasets frequently contained the number of people *infected*, *recovered* or *dead* due to COVID-19 with regard to the *country* and occasionally the progression of these quantities over *time*.

In this lab we will focus on displaying data in meaningful ways. Besides its quantitative nature this task has some qualitative properties. Similar to many solutions of problems in computational geometry, for instance, deciding whether a point is contained in a polygon or the construction of a convex hull for a set of points, the quality of a data plot is obvious to the human observer. Not so for a machine.

Consider the two figures shown below. While Fig.**??** is highly informative for humans such as policy makers, Fig.**??** bears little information content. Both figures, however, were generated by the same depiction algorithm, processing data from the spread of COVID-19 during 2020. The algorithm can be reviewed in the appendix.



(a) 23.02.2020 (b) 11.05.2020

Figure 1: Comparison of a typical scatterplot depicting the $logarithm$ of confirmed cases as the radius.

For now, we start by defining quantitative measures, i.e. measures of data visibility that can be understood by a computer and assessed on a purely mathematical basis. A discussion on how well the developed algorithms and approaches hold up from a qualitative viewpoint will be conducted later on.

Fundamental work into the quantification of visual quality was done both by Tufte in his 1983 book, *The visual display of quantitative information graphics press* and Miller et al. in *The Need For Metrics In Visual Information Analysis*. Based on this, follow-up work and further considerations regarding visibility problems and sorting we develop a novel approach to visualize multidimensional data in a scatterplot designed to transport as much information as possible.

Our main resource will be the paper by Sergio Cabello et al. *Algorithmic Aspects of Proportional Symbol Maps*. In it, the authors discuss the difficulty of maximizing utility measures with regards to so-called "physically realizable drawings", in particular they prove this problem to be NP-hard. As a consequence, they restrict their further discussion on a distinct subset of physically realizable drawings, called "stacking drawings" which are determined solely by a total ordering of the symbols, called the "stacking order". They then derive algorithms that determine optimal stacking orders in polynomial runtime. Those algorithms are designed for datasets where each *glyph* only depicts a single quantity.

Our goal is to generalize these algorithms so that they may be applied to other types of *glyphs*. We focus in particular on glyphs that visualize multiple quantities, as is needed for data relating to COVID-19, such as nested discs, centered nested discs and pie charts.

2 Problem Statement

Cabello et al. discussed proportional Symbol Maps. Such a map consists of a finite set of 2-dimensional figures called symbols or glyphs. Each of these glyphs has a position in $\mathbb{R}^2$ and a radius or size corresponds to some quantity given by the data it is supposed to represent. A classic example is the representation of earthquakes and their strength on the Richter-scale. Opaque discs are centered at epicenter of the earthquake, the radius corresponds to its strength. In this case, the glyphs are the discs and the associated data encoded in the radius indicates the strength. Other types of glyphs, like squares, hexagons, or pie charts are also possible and are investigated in this lab.

The problem is to find a drawing which retains as much of the information about the quantity and the location of the center as possible. More concretely, the raw data alone does not tell you how to deal with overlapping discs, squares or occluded pie pieces. In order to visualize the data on a map we must make several decisions, e.g. one pie occludes a pie segment of a lower pie. Should we change the order? Can we rotate the lower pie s.t. all of its segments are visible?

For the definition of a drawing Cabello et al. describe two approaches, *physically realizable drawings* and *stacking drawings*. For our purposes, an intuitive definition of what is talked about will suffice. Those interested in a more detailed definition can consult the original paper.

Intuitively, a physically realizable drawing can be imagined as having originated by taking very flexible glyphs, arranging them in $\mathbb{R}^3$ such that they do not intersect (potentially in a Dali-like fashion) and then taking a picture from above.
Similarly, a stacking drawing is the same thing but the glyphs are rigid and can only have the same height for each point on the same glyph.

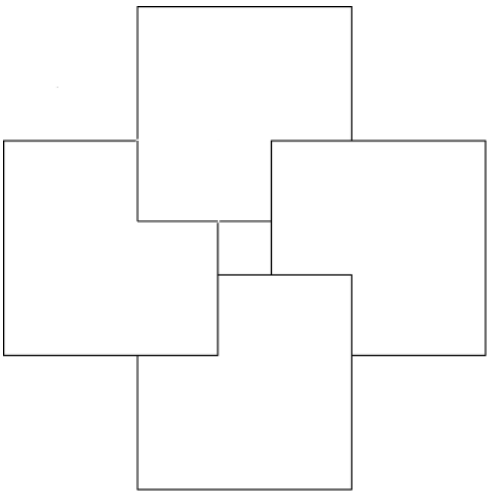


Figure 2: A physically realizable drawing of squares which is not a stacking drawing

Figure **??** gives an example of a physically realizable drawing which is not a stacking drawing. The squares overlap cyclically going clockwise "upwards" reminiscent of the "Penrose Stairs". This image can not occur if the squares where embedded with a fixed height as it is required for stacking drawings.

A stacking drawing is uniquely determined by the order (ascending) in which the glyphs are stacked on top of one another. This is the so-called *stacking order* of the drawing. Formally, given a finite set D of n glyphs, a stacking order is a bijection $\phi : [n] \to D$. This bijection describes the order in which the glyphs are drawn. Glyphs which are drawn later may cover parts of the glyphs which are drawn earlier. Therefore our task is to find a stacking order such that as much as possible of every glyph is visible.

We are yet to define what we mean when talking about a glyph being as visible as possible. Since this may depend on the type of glyph considered we will turn our attention towards circular discs for now. The other definitions will be delivered as soon as they are needed. We refer to these quantitative measure for the visual quality as *global utility functions*.

We take the perspective of a viewer looking at the map. The viewer should be able to identify the size of the circles and their center points. To achieve this goal, we could try to maximize the visible area of each circle, but there are cases in which that would not allow the viewer to get all of the information. For instance, if the perimeter is not visible at all, then neither the location of the center point nor radius can be determined exactly. The approach we will use is to maximize the visible perimeter of our discs.

For a fixed stacking order ϕ, we will denote the disc $\phi(i)$ as D_i. We then define the visible and occluded boundary for discs formally by

$$U_i^{vis} = \partial D_i \setminus \bigcup_{j>i} D_j$$
$$U_i^{occ} = \partial D_i \cap \bigcup_{j>i} D_j.$$

furthermore, we denote the length of the visible perimeter as $|U_i^{vis}|$. Two global utility functions come to mind:

$$\lambda_{min}(D) = \min_{i\in[n]} |U_i^{vis}|$$
$$\lambda_{sum}(D) = \sum_{i=1}^{n} |U_i^{vis}|$$

While λ_{min} is focused on the least visible perimeter, λ_{sum} sums up the visible length of all perimeters. Key results proven in *Algorithmic Aspects of Proportional Symbol Maps* are the following

Theorem 1. *It is NP-hard to decide if a given collection of congruent disks has a physically realizable drawing where at least some given length of the perimeter of each disk is visible.* □

This implies that trying to maximize the minimal $|U_i^{vis}|$ with respect to physically realizable drawings is NP-hard.

Theorem 2. *It is NP-hard to decide if a given collection of disks has a physically realizable drawing whose total visible perimeter is at least a given value.* □

This implies that trying to maximize the sum over all the $|U_i^{vis}|$ with respect to physically realizable drawings is NP-hard.
For stacking orders they give a positive result with regards to the minimum:

Theorem 3. *Given n disks in the plane, a stacking order maximizing the boundary length of the disk that is least visible can be computed in $O(n^2 \log n)$ time.* □

Whether a polynomial algorithm for the sum of all visible perimeters exists remains an open question.

We won't attempt to construct an algorithm for the sum problem and instead take a look at glyphs which are able to depict more than one piece of information.
The first glyph we will discuss are nested discs. Formally, every glyph G^i is given by a set of discs $G^i_1, ..., G^i_k$ such that $G^i_{j+1} \subset G^i_j$ for all $j \in \{1, ..., k-1\}$. Often times figures using nested discs also center all the discs in the same point.
The second type of glyphs we will use are pie charts. A pie chart is formally given by a disc D_i and a set $\{\alpha^i_1, ...\alpha^i_k\}$ of dividing lines which we can encode as angle in $[0,\ 2\pi]$.

In the next section we will discuss a simple and versatile approach to determine optimal stacking orders for glyphs maximizing utility functions with specific properties for glyph stacking order drawings and then apply it to our glyphs and some utility functions.

3 Stacking order algorithms for more general Glyphs

The function λ_{min} from above provides an example of a global utility function for which an optimal stacking order can be computed via a simple greedy algorithm. At each time step just calculate for all discs their visible perimeter as if they were the bottommost disk, choose the disc D_i with the largest visible perimeter and put it on the bottom. In the next timestep we look at the set of disks without D_i and repeat the procedure. This is done until we have processed all disks.

We can now formalize this approach for more general objects and prove that the greedy algorithm is optimal with respect to the utility.

Let D be a finite set and for $d \in D$ and $S \subseteq D$ the function $\Gamma(d, S) \mapsto \mathbb{R}_{\geq 0}$ with the property, that for every $d \in D$ we have $S' \subseteq S \implies \Gamma(d, S') \geq \Gamma(d, S)$.
The set D corresponds to our set of glyphs and Γ corresponds to a local utility function, which depends on the insertion order, i.e, the set S is thought of as the set of all glyphs which lie above d. Let $\phi : \{1, ..., n\} \mapsto D$ be a stacking order. Then the for every glyph $d_i := \phi(i)$ the set of glyphs above d_i is given by $\{d_j \in D \mid j > i\}$.

```
1  GreedyStacking(finite set D)
2    x := argmax_{d∈D} Γ(d, D \ d)
3    return [x, ALG(D \ x)]
```

Figure 3: The greedy stacking algorithm

Theorem 4. *For a given set D and a local utility function Γ the greedy Algorithm as described in* **??** *returns a stacking order s which maximizes*

$$\lambda(s) = \min_{i\in\{1,\dots,n\}} \Gamma(s(i), \{s(j) \mid j > i\}).$$

Proof. To prove the above theorem we define s^* the stacking order generated by the algorithm. We will then prove that for any other order $s \neq s^*$, there is a modified order s' that fulfills $\lambda(s) \leq \lambda(s')$ and that overlaps with s^* in at least one more additional spot. Consider $s \neq s^*$ and let $i = \min\{l \mid s(l) \neq s^*(l)\}$ the first index for which s and s^* disagree. The glyph $s^*(i)$ has a different index according to s, namely, the index is $j = s^{-1}(s^*(i))$ and, because i is the minimal index of disagreement, we have $j > i$.
We modify s as follows. Instead of choosing glyph $s^*(i)$ only in index j, the new stacking order s' selects it at i as well and increases the index of $s(l)$ by 1 for $l \in [i, ..., j-1]$. More formally, $s' = s \circ \tau$, where

$$\tau(k) = \left\{ \begin{array}{ll} k-1, & \text{for } k \in [i+1, \dots, j] \\ j, & \text{for } k = i \\ k, & \text{else} \end{array} \right\}$$

Clearly, $s'(i) = s \circ \tau(i) = s(j) = s^*(j)$ and for $k < i$, $s'(k) = s \circ \tau(k) = s(k) = s^*(k)$, so s' disagrees with s^* only at a later stage. We claim that s' is as least as good of a solution as s, meaning $\lambda(s') \geq \lambda(s)$. If this holds true we can, by iterative application of the previous modification, construct for any stacking order s a non-decreasing sequence $s, s', s'', \dots, s^*$ ending in s^*, the stacking order generated by the algorithm. Thereby proving that s^* is optimal.
The proof of the claim rests on the following observation: Recall that the Γ-value is anti-monotonous, meaning its value increases as the second argument decreases. All the objects that either maintain the same index or get it increased by one only have a subset of glyphs above them as compared to before the modification, meaning their values increase. Only the glyph whose index changes from j to i has a growing set of glyphs above it. However, this object is selected by the algorithm at index i so it has to be at least as good as the glyph selected by s. Thus, the utility of each disk as arranged in s' is bounded below by the utility of some disk as arranged in s.
To make this formal, we prove the following:

$$\forall k' \in [n]\ \exists k \in [n] :\ \Gamma(s'(k')|\{s'(l), l > k'\}) \geq \Gamma(s(k)|\{s(l), l > k\}) \tag{1}$$

Consider first $k' \notin \{i, \dots, j\}$, then

$$\Gamma(s(k'), \{s(l),\ l > k'\}) = \Gamma(s'(k'), \{s'(l),\ l > k'\})$$

because both $s(k') = s'(k')$ and $\{s(l),\ l > k'\} = \{s'(l),\ l > k'\}$. We can choose $k = k'$.
If $k' \in [i+1, \dots, j]$, then

$$\Gamma(s'(k'), \{s'(l),\ l > k'\}) \geq \Gamma(s'(k'), \{s(l),\ l > k'-1\}) = \Gamma(s(k'-1), \{s(l),\ l > k'-1\})$$

where the inequality comes from the second argument on the left-hand side being a subset of the one on the right, so $k = k' - 1$ and lastly for $k' = i$

$$\begin{aligned}\Gamma(s'(k'), \{s'(l), l > k'\}) &= \Gamma(s'(i), \{s'(l),\ l > i\}) \\ &= \Gamma(s^*(i), \{s^*(l),\ l > i\}) \geq \Gamma(s(i), \{s(l),\ l > i\})\end{aligned}$$

Here, the first equality is simply substituting $k' = i$. The second one comes from s' and s^* agreeing for $l \leq i$ and the inequality holds because $s^*(i)$ is the choice of the algorithm and thus maximizes precisely this term, so $k = i = k'$ does the job.
In total

$$\begin{aligned}\lambda(s') &= \min_{i \in \{1,\dots,n\}} \Gamma(s'(i), \{s'(j)\ | j > i\}) \\ &= \Gamma(s'(k'), \{s'(j) \mid j > k'\}) \text{ for some } k' \\ &\geq^{(1)} \Gamma(s(k), \{s(j) |\ j > k\}) \text{ for the corresponding } k \\ &\geq \min_{i \in \{1,\dots,n\}} \Gamma(s(i), \{s(j)\ | j > i\}) = \lambda(s)\end{aligned}$$

proving that s' is as least as good a solution as s. □

We can use this theorem for glyphs of any kind, in particular for the glyphs we discussed before, so long as the utility function can be modelled as described in the theorem.

With this, all we need to do is to define meaningful utility functions for our different glyphs and evaluate them locally with feasible runtime.

3.1 Centered nested disks

For centered nested disks we can easily define local utility measures. Every glyph G corresponds to k nested circles therefore we could just calculate the visible perimeter for the k circles and then just combine them by choosing the minimum of the k circles or the sum of the visible perimeter of the k circles. For some set S of objects above G_i we then get formally:

$$\Gamma_{min}(G^i, S) = \min_j {}^{vis}G^i_j$$

$$\Gamma_{sum}(G^i, S) = \sum_{j=1}^{k} {}^{vis}G^i_j$$

We may also be interested in the relative utility which is given by normalizing the size of the circles. This relative approach stops the small circles from dominating big circles. In our applications there is close to no difference between the relative and absolute utility.

For Γ_{min} it may happen that for all glyphs the smallest circle may be completely covered. Then the utility would be zero for all of the circles. If that happens the algorithm

has an optimal solution of value zero. If that happens, we perform the greedy step again and ignore the smallest circle in all of the glyphs until we get a value bigger than zero. This heuristic gives better results than just choosing a random circle.

Remark. *It is NP-hard to calculate physically realizable drawing where* $\min \Gamma_{min}$ *or* $\min \Gamma_{sum}$ *is maximized since disc glyphs are a special case of nested discs and both utility measures coincide and are equal to the visible circumference.*

We are primarily interested in the quality of the optimization, not the runtime of the computing algorithms. Therefore we use a naive implementation to calculate the stackings with runtime $O(n^3k^2)$. The runtime could be improved to $O(n^2k \log nk)$ by adapting the data structure introduced in [**?**].

3.2 Nested disks

For dense regions centered nested discs loose a lot of information since the small circles can easily be completely covered up. Therefore we will now derive an algorithm which guarantees that every nested circle is at least visible. To achieve that we will allow the circles to be moved within each other. We still want to have $G^i_{j+1} \subset G^i_j$ for all j but the centers don't have to coincide.

Our approach is quite simple. We first just look at the biggest circle of each glyph. For those we perform our greedy algorithm with the local utility given by the size of the maximal *continuous* visible perimeter. After we have done that we can take the point p on the circle which is in the middle of the longest continuous perimeter piece visible and connect it to the center of the big circle by a line. Now every center point for a nested inner circle is uniquely defined by positioning it on that line such that the circle touches the point p. The final glyphs look a little bit like the *Hawaiian earring* topological space.

The resulting drawing guarantees that the perimeter of every circle is at least a little bit visible. The downside of this construction is that the drawing look a little bit strange to the viewer and its more difficult to focus the center of a glyph.

We again use the naive approach for disks from [**?**] with runtime $O(n^3)$. Since we are only interested in the largest disk the number of nestings k only adds a linear term. Therefore the overall runtime is given by $O(n^3 + nk)$. As described in [**?**] the runtime could be improved to $O(nk + n^2 \log n)$.

3.3 Pie charts

Recall a pie chart glyph is given by a disk D_i and dividing lines given by a finite set $P_i = \{\alpha^i_1, ...\alpha^i_m\}$. The α_j are given by angles in $[0, 2\pi]$. We may rotate the dividing lines

like minute or hour hands on a mechanical watch, but the relative distance between all angles must stay the same. This rotation can be described by a single angle due to the fixed relative distance of the dividing lines. Given a rotation of the angles and a set S of glyphs which lie above the pie chart in our drawing, we can define $\Gamma_{pie}(D_i, S)$ as the minimal radial distance of any of the α_j^i to the covered perimeter. If the perimeter is completely visible, then this distance is defined as half the circumference or πr_i.

We now need to investigate two things:

1. Is it feasible to optimize the aforementioned utility function with respect to physically realizable drawings generally or do we need to restrict ourselves to stacking drawings? The pie chart glyphs are not a natural generalization of the disks. Therefore we cannot argue NP-hardness as easily as in the case of nested disks.
2. In addition to deciding which pie covers which we also have to specify a rotation for each pie. How do we calculate the optimal rotation for a given pie knowing which pies lie above it.

3.3.1 NP-hardness

Setting

Given a finite set D of disks d_i in the plane with centers c_i and positive radii r_i together with a finite set $P_i = \{\alpha_1, \ldots, \alpha_{m_i}\}$ of points on the boundary of d_i subdividing the disk into pie-pieces, we wish to determine a physically realizable drawing of D together with a rotation φ_i for each disk, s.t. the minimum distance of any point in $p \in \bigcup P_i$ to the next occluded point on the boundary of d_i is maximized. I.e. we are given Pie-Charts and would like to arrange and rotate them in such a manner that the points where the transition-lines of the pie-chart meet the boundary are as far from being occluded as possible.

Theorem

It is NP-hard to decide, whether for a given set D of disks with boundary points P_i there is a physical realization (with rotation), s.t. the minimum distance of any of the distinguished points p_i to the next occluded point along the perimeter of its perimeter is greater or equal to k. As a formula:

$$\min_{i \in [n]} \Gamma(D_i, \{D_j \mid j > i\}) \geq k$$

.

Proof

The proof is inspired by and in large parts analogous to the proof of Theorem 1 [**?**] concerning. We aim to reduce the NP-hardness from $planar$-3-SAT, i.e. instances $\mathcal{I}$ of

3-SAT with the property to that graph $G(\mathcal{I})$ is planar. The graph $G(\mathcal{I})$ is bipartite, has a node for each variable on one side and each clause on the other. A variable node is connected to the clause iff one of it's literals appears in the clause.

Example:

$$\mathcal{I} = \underbrace{(x_1 \vee x_2 \vee \overline{x_3})}_{C_1} \wedge \underbrace{(x_3 \vee \overline{x_4} \vee x_5)}_{C_2}$$

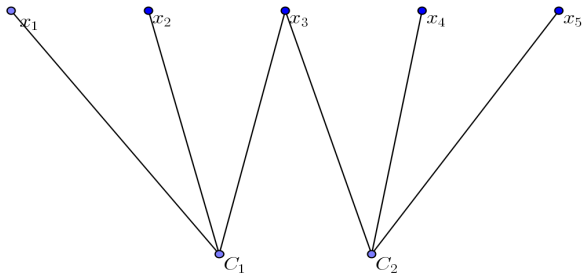


Figure 4: The incidence graph $G(\mathcal{I})$

Given a planar instance $\mathcal{I}$ of 3-SAT we need to construct an instance $\mathcal{D}$ of our pie-charts, s.t. $\mathcal{I}$ is satisfiable iff $\mathcal{D}$ has a physical realization with distance $\geq k$ for some suitable k.
All the disks presented have the same radius and we say that two disks overlap by f if the overlapped fraction of the boundary has length f. We introduce the construction of $\mathcal{D}$.

Construction

We use the three disk types in Figures **??** and **??** to construct for each variable in the 3-SAT instance a so-called "gadget". A gadget is a group of neutral disks forming an elongated band, each overlapping the next by a fraction of 1/4 or 90 degrees. Notice that the maximum possible distance of separation points to occluded area is 15 degrees, because the segment is 120 degrees and 90 are covered. The gadget is constructed in such a way that it can only be oriented clockwise or counter-clockwise if we want $k > 0$ separation between the lines to the occluded area.

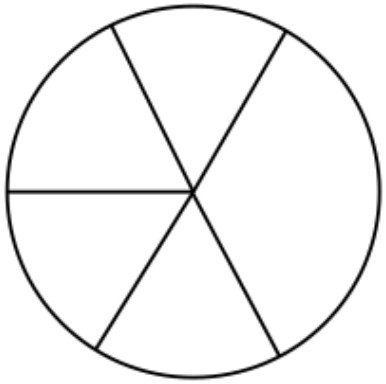


Figure 5: A "neutral" disk used as a connector within a variable ring. The angles are 60 and 120 degrees

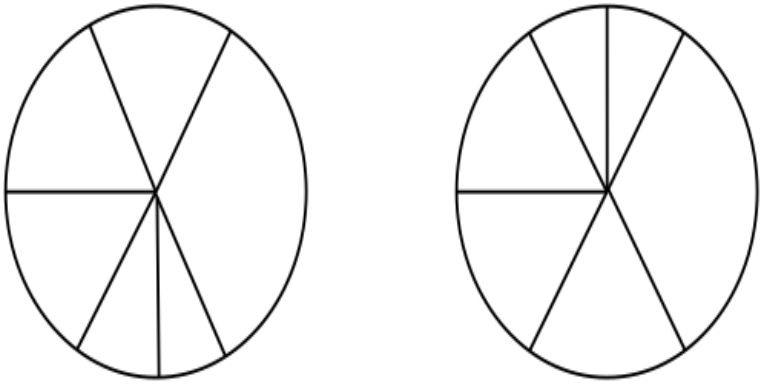


Figure 6: A "true" and "false" disk, similar to the neutral one above, one of the 60 degree segments is cut in half

Figure **??** depicts a prototype of a gadget, all the disks are of the neutral type for now. The curvature is created by connecting the disks at a 45 degree angle while still overlapping by a fraction of 1/4.

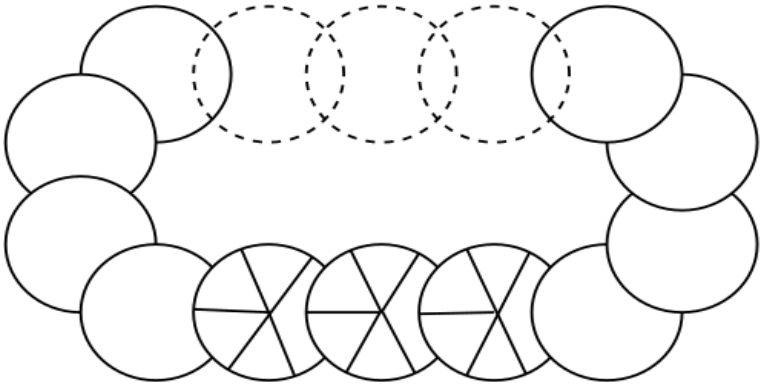


Figure 7: One such gadget G_i for each variable x_i

We will interpret a gadget being ordered ascending clockwise as the corresponding variable being "true" and otherwise "false".
For each occurrence of the variable x_i in any of the clauses we will, in the gadget G_i substitute a neutral disk by either a "true" or "false" disk depicted in Figure **??**. We

use a "false" disk if the literal appears negated in the clause.

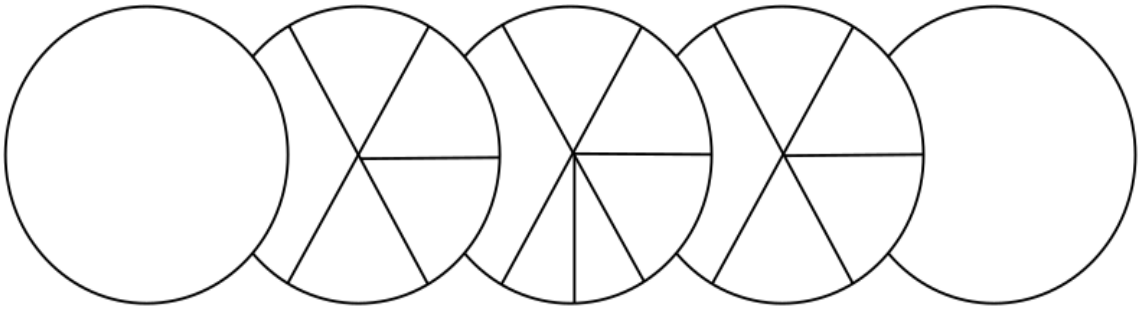


Figure 8: A segment of a "true" gadget with a "false" disk in it

The above segment is of a "true" gadget as the disks are clockwise ascending. The "false" and "true" disks will ensure that the added segment will point outwards iff the corresponding literal in the clause is false.
Next we will focus on how the clauses will be set up. Later we will connect gadgets and clauses, which is why we require the instance to be planar.

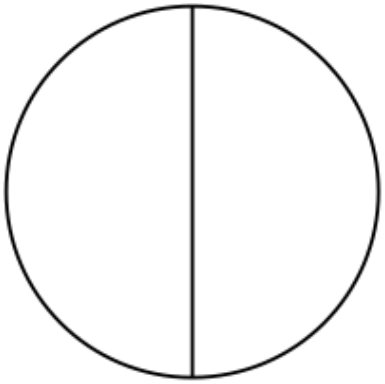


Figure 9: One such disk for each clause

Recall that the instance is satisfiable iff each clause is fulfilled. We instantiate one disk as above for each clause. It has two separating lines exactly opposing one another. For each literal in the clause we overlap it with one of the disks show below by a fraction of 1/4 as shown in Figure **??**.
This connector disk is supposed to be above the clause-disk iff its associated literal is "false" in the clause.
The clause disk is overlapped by three connector disks, each overlapping 90 degrees and leaving three 30 degree corridors. If the upper connector disk can be rotated as depicted, then it may go under the clause disk, signifying the literal is "true" in the clause. Otherwise it must go over, like the bottom left one. If all three connectors go over the clause, then the clause disk cannot have 15 degrees of space between the separating points and the occluded area. If, conversely, at least one of them goes under, like potentially the top one in Figure **??**, then by rotating the clause disk as depicted creates 15 degrees of length between the lines and the occluded area.
Now we need to bring it all together. Recall that we have gadgets that can be oriented

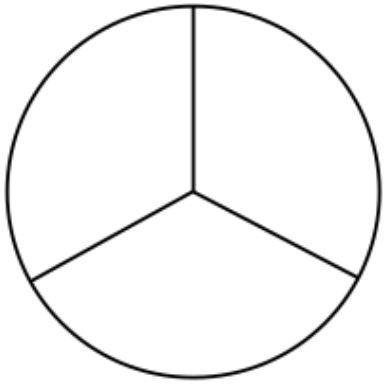


Figure 10: This disk is the endpoint of the edge connecting the gadget to the clause disk

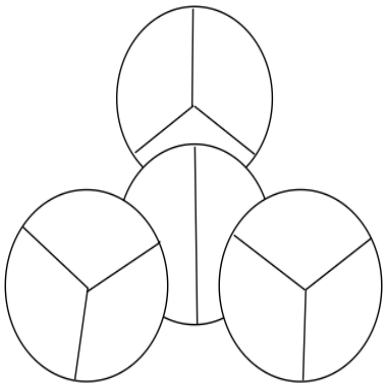


Figure 11: The clause together with the literal connectors

clockwise or counterclockwise to signify a "true" or "false" assignment of the variable. The gadgets have for each variable-clause pair a special "true" or "false" disk that will point a separating line outwards iff the *literal* is false in the clause. We also have connectors at the clause that will act as if "false" iff they go over the clause disk, which has to occur if their separating line cannot be arranged as in the top disk of Figure **??**.
What we need to ensure is that the connector disk cannot be rotated like that if the literal disk in the gadget points outwards. To do this we introduce the last building block of the construction.

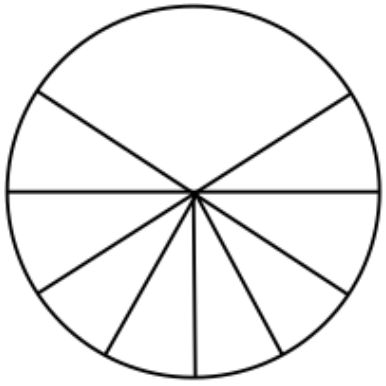


Figure 12: These disks connect the gadgets to the corresponding clauses

These disks will be arranged in a line connecting the "true"/"false" disks in the gadget with the clause-connector disks at the clause like so:

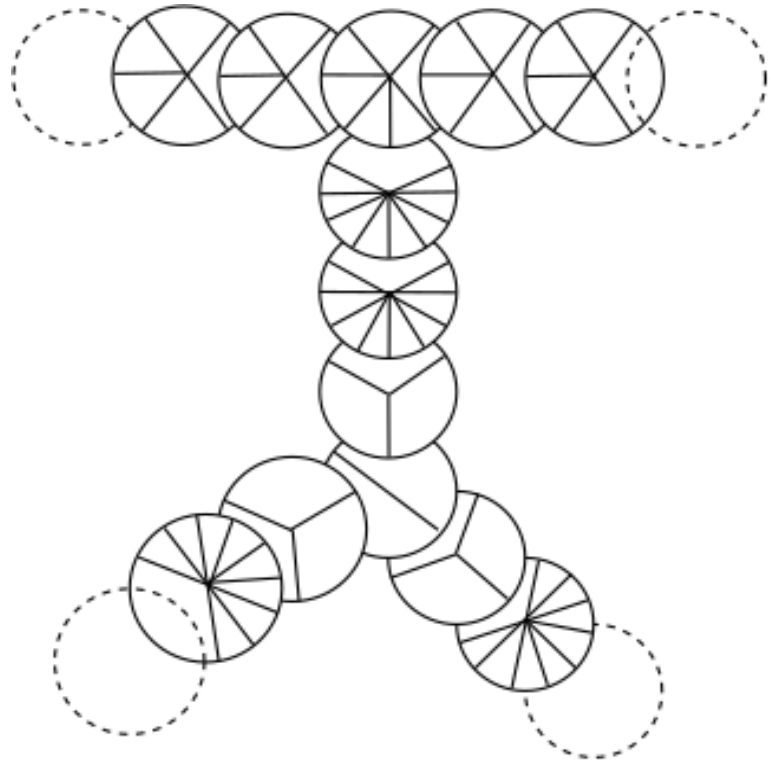


Figure 13: These disks connect the gadgets to the corresponding clauses

The top gadget is oriented as "false" and the "true" disk in the middle thus has a separator pointing outward. This forces the first outgoing connector, which overlaps the "true" disk by 1/12 = 30 degrees to be under it and point its own separators downwards as well. This continues until the clause-connector disk is reached which now cannot be oriented as the bottom-right own and has to go over the clause-disk.

Example

Example will be depicted here.

Conclusion

The logical properties were already explained when motivating the construction but will be formalized here again. Notice that the disks in the clause and the gadget can create a distance of 15 degrees between occlusion and separation points. We claim the following. Given a planar instance of 3-SAT $\mathcal{I}$ and the corresponding construction $\mathcal{D}$ as above, $\mathcal{I}$ is satisfiable if and only if there is a physical realization of $\mathcal{D}$ that creates a distance of at least 15 degrees $= (15/360 \cdot 2\pi)$.
Given a satisfiable instance we orient the gadgets according to the assignment. This will make all "false" connections point a separator outward. Within the gadgets, all disks maintain a distance of at least 15 degrees to the other disks in the gadget. The

connector disks at these "false" connections will have to go under so that the outpointing separator has distance ≥ 0, let alone 15. The "true" connections are not forcing the outgoing connections to go under, we will place them above giving a distance of exactly 15 degrees. This connection line will be going "upwards" (as viewed from gadget to clause) until it reaches the clause connector, which now can be placed under the clause-disk enabling it to also create a distance of at least 15 degrees. This implies that all disks can create a distances of 15 degrees to their occluded boundary segments.
Conversely, if the constructed instance has a physical realization with degrees ≥ 15, then clearly in each clause-disk we must have a least one of the connector going below it. This is only possible with distance ≥ 15 if the separators are directed away from the clause disk and thus forcing the rest of the connection to go "upwards" (as viewed from clause to gadget). This means that the first disk in the gadget must be under the last disk in the connecting line. This is only possible if the literal is "true" meaning the variable is "false" and the literal negated or "true" and the literal in normal form. We obtain and orientation of the gadgets that is consistent with all clauses and thus a satisfying assignment of the variables in $\mathcal{I}$.□

3.3.2 Optimal rotations for Γ_{pie}

Given a set S of covering glyphs and a fixed pie chart C_i we can calculate the occluded radial intervals $I_1, ..., I_k$ on the boundary circle of C_i. We now want to find a rotation of the dividing lines that maximizes the minimal distance of any of the α_j^i to the occluded intervals. We can look at this rotation with respect to some reference line. Here we use α_1 as the reference line.

Now we want to derive all of the positions on the circle where the reference line can be positioned such that none of the dividing lines are covered on the boundary. We will derive the complementary intervals first, i.e., all the intervals that guarantee that at least one dividing line is occluded.

Lemma 1. *Given a set of occluded intervals $I = \{[a_1, b_1], ...[a_k, b_k]\}$ (as angles) and a set of dividing lines $\{\alpha_1, ...\alpha_m\}$ also given by angles, let β_j be the counterclockwise angular distance from α_1 to α_j for $j \in \{1, \ldots, m\}$, and let $\mathcal{I}$ be the set system given by*

$$\mathcal{I} = \bigcup_{i=1}^{m} \bigcup_{j=1}^{k} \{[a_j - \beta_i, b_j - \beta_i]\}.$$

Then any rotation which locates α_1 in one of the intervals in $\mathcal{I}$ occludes at least one dividing line. Conversely, any rotation locating α_1 in none of the intervals in $\mathcal{I}$ will occlude none of the lines.

Proof. " $\Longrightarrow$ ": Consider a rotation β, s.t. $\alpha_1 + \beta \in \mathcal{I}$, then $\alpha_1 + \beta \in [a_j - \beta_i, b_j - \beta_i]$ for some pair (i, j). Note that $\alpha_1 + \beta_i = \alpha_i \implies \alpha_1 = \alpha_i - \beta_i$. Therefore, $\alpha_i - \beta_i + \beta \in [a_j - \beta_i, b_j - \beta_i]$

and thus $\alpha_i + \beta \in [a_j, b_j]$. The line represented by α_i will be occluded.
" $\Longleftarrow$ ": Analogously. □

The complement of $\mathcal{I}$ on the circle describes all of the locations of α_1 which do not occlude any dividing line. This complement may be empty since there may not be a rotation which does not occlude any line. With this observations we now can easily derive an algorithm which returns a rotation which maximizes our utility.

Theorem 5. *Given a set of occluded intervals $I = \{[a_1, b_1], ...[a_k, b_k]\}$ (as angles) and a set of dividing lines $\{\alpha_1, ...\alpha_m\}$ also given by angles, Algorithm 2 returns a rotation which maximizes the minimal radial distance of any of the α_j^i to the covered perimeter.*

Proof. Algorithm 2 rotates a given dividing line in such a way, that it lies exactly in the midpoint of the biggest not occluded interval $[a_v, b_v]$. The midpoint in an interval is the point which has the maximum distance from both boundaries, a_v and b_v. As the biggest not occluded interval maps to the circle's perimeter, it also maximizes the distance to the boundaries of the occluded intervals $\{[a_1, b_1], ..., [a_k, b_k]\}$. □

---

1 Input: $I = \{[a_1, b_1], ...[a_k, b_k]\}$ $\{\alpha_1, ...\alpha_m\}$)
2 calculate $\mathcal{I} = I \cup \bigcup_{i=2}^{m} \bigcup_{j=1}^{k} \{[a_j - \beta_i, b_j - \beta_i]\}$
3 let $\mathcal{V}$ be the set of maximal continuous intervals in $[0, 2\pi] \setminus \mathcal{I}$
4 $v = [a_v, b_v]$ biggest interval in $\mathcal{V}$
5 $r = (b_v + a_v)/2$
6 return r

Figure 14: IntervalMidpointAlgorithm

With this algorithm we can again use our general greedy algorithm for stacking orders.

If we use a naive implementation for Theorem 5 we get a runtime of $O(n^3k^2)$. Again the data structure introduced in **[?]** could be used for the pie glyphs which yields a runtime of $O(n^2k \log nk)$. This adaptation would need some work, since the pie glyphs are not immediately compatible with the data structure.

It may happen again that for all glyphs the local utility is zero. Again we have to use a heuristic to get better results. For instance, we can just ignore one dividing line for each glyphs and try again. This can be done iteratively until the utility is not zero for all glyphs. For the case of exactly one dividing line the utility must be bigger than zero and therefore the heuristic guarantees a better decision than just randomly tie breaking.

3.4 Squares

The last glyph we discuss are squares which are subdivided into smaller rectangles. We don't restrict ourself to axis aligned squares. Therefore the algorithms we discuss may rotate the squares to get better results with respect to the visibility.
Cabello et al. only discussed axis aligned Squares in [**?**]. There hasn't been any theoretical discussion with regards to rotatable squares and visibility as far as we know. Our greedy stacking theorem would not apply to rotatable squares since the local utilities may only depend on the order, but the rotation we perform at time step t could change some of the local utilities for squares which have been drawn before time t.
For that reason we would need some new approach to get any optimal results and it may even be NP-hard to calculate a stacking with optimal rotations with respect to the visibility of the perimeter of the squares.

Our main focus in this Lab was the visualisation of multiple features, e.g. in the case of covid-19 infected,dead and recovered. Consequently we do not try to derive optimal algorithms for the single feature case and just recycle some of our other results to get a heuristic solution for our problem with respect to squares.
We now derive a data representation in the form of squares and sub-rectangles for exactly 3 features. Our glyph will be given by a Square S, such that the length of the sides represents the sum of the features with respect to some scaling. Then we insert an orthogonal dividing line l_1 which divides the square into two rectangles such that one of the rectangles r_1 contains an area proportional to the largest feature. With a second line l_2 the other rectangle is divided into to rectangles r_2 and r_3 which each contain an area proportional to the other features. See Figure 15 for an example of such a glyph.

In order to guarantee that the user can get all of the depicted information, he needs to see to diagonal corner points of the square and one point on the border for each line. We chose the point p_1, p_2, p_3 and p_4 which are displayed in Figure 15 for our algorithm. The local utility of a square would be the minimal distance of the points to the occluded perimeter.
We can get a good solution with the help of pie charts. For every square we construct a

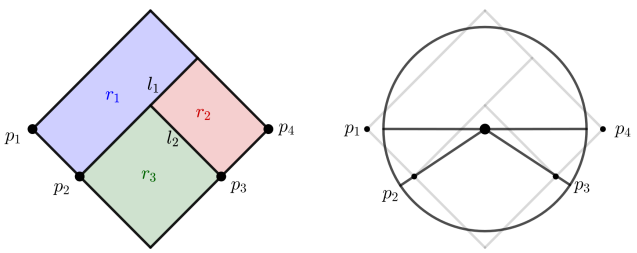
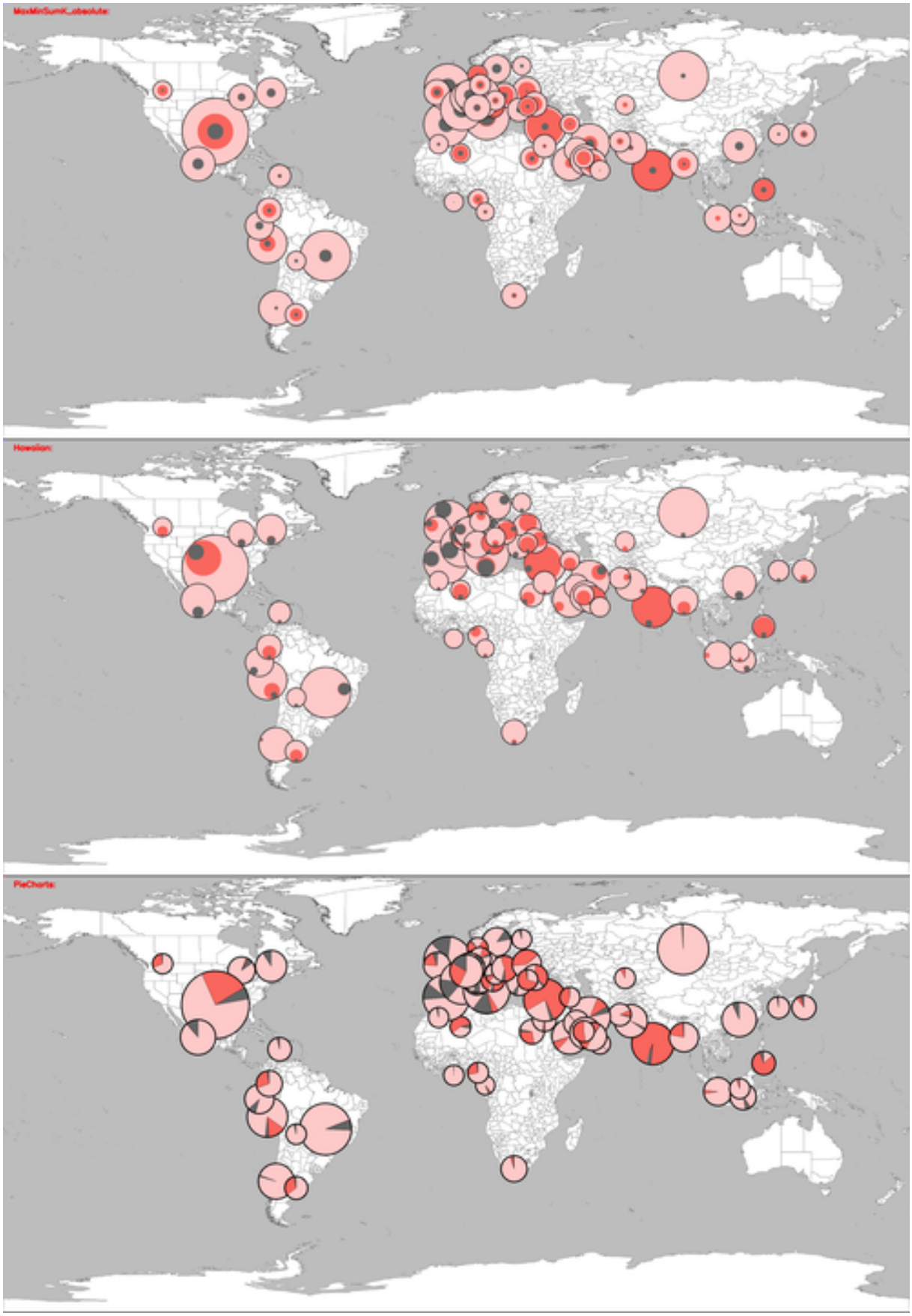


Figure 15: left:square glyph, right: heuristic pie chart

pie chart centered in the square. The dividing lines correspond to the points p_i and the radius is given by the average distance of the points p_i to the center point. Again this construction is depicted in Figure 15. Our algorithm performs the pie chart algorithm for the heuristic pies and then applies the stacking and the rotations to the squares.

4 Experimental validation


MaxMinSumK_absolute
Hexalon
PieCharts

References

[1] Tufte, Edward R. "The visual display of quantitative information graphics press." Cheshire, Connecticut (1983).

[2] Miller, Nancy, et al. "The need for metrics in visual information analysis." Proceedings of the 1997 workshop on New paradigms in information visualization and manipulation. 1997.

[3] Brath, Richard. "Metrics for effective information visualization." Proceedings of VIZ'97: Visualization Conference, Information Visualization Symposium and Parallel Rendering Symposium. IEEE, 1997.

[4] Tatu, Andrada, et al. "Visual quality metrics and human perception: an initial study on 2D projections of large multidimensional data." Proceedings of the International Conference on Advanced Visual Interfaces. 2010.

[5] Urribarri, D. K., & Castro, S. M. (2016). Prediction of data visibility in two-dimensional scatterplots. Information Visualization, 16(2), 113–125. doi:10.1177/1473871616638892

[6] Coeurjolly, D., Miguet, S., & Tougne, L. (2004). 2D and 3D visibility in discrete geometry: an application to discrete geodesic paths. Pattern Recognition Letters, 25(5), 561–570. doi:10.1016/j.patrec.2003.12.002

[7] Yang-Pelaez J and Flowers WC. Information content measures of visual displays. In: Proceedings of the IEEE symposium on information vizualization 2000 (INFOVIS'00), 2000, pp. 99–103. Washington, DC: IEEE Computer Society.

[8] Sergio Cabello, Herman Haverkort, Marc van Kreveld, Bettina Speckmann. Algorithmic Aspects of Proportional Symbol Maps. In: Algorithmica (2010) 58: 543–565. Published with open access at Springerlink.com.

———————-