GeomLab 2020

October 12, 2020

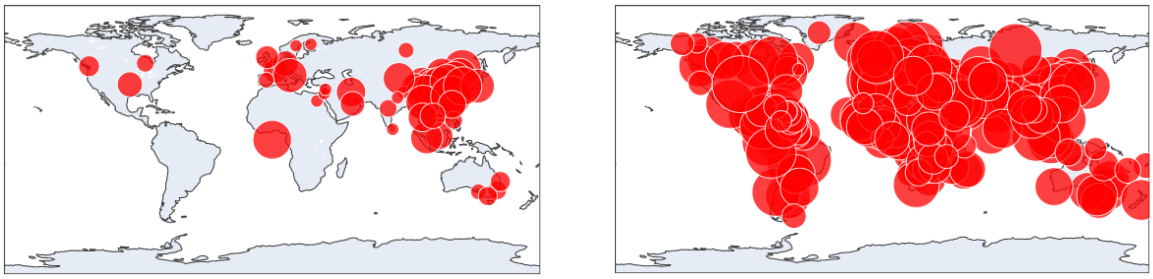
Contents

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. 1a is highly informative for humans such as policy makers, Fig. 1b 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 scatter plot 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 disks, centered nested disks 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 corresponding 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 disks are centered at the epicenter of the earthquake, while the radius corresponds to its strength. In this case, the glyphs are the disks 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 as possible. More precisely, the raw data alone does not tell you how to deal with overlapping glyphs like disks, squares or covered pie pieces. In order to visualize the data on a map we must make several decisions, e.g., one pie covers a pie segment of a lower pie. Should we change the order? Can we rotate the lower pie such that 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 [8].

Intuitively, a physically realizable drawing can be imagined 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 birds eye view. Similarly, a stacking drawing is the same thing with the glyphs being rigid and having the same "height" in a dimension perpendicular to the surface being looked at from birds eye view for each point on the same glyph.

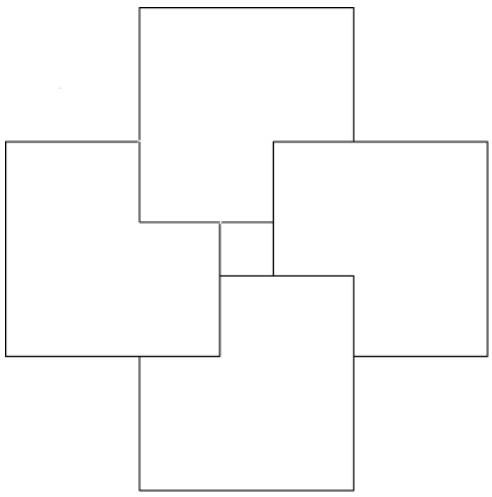


Figure 2: A physically realizable drawing of squares which is not a stacking drawing

Figure 2 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 (ascending) order 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 "much visible" as possible. Since this may depend on the type of glyph considered we will turn our attention towards circular disks 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 function*.

We take the perspective of an onlooker looking perpendicularly 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 glyphs being disks in the following.

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

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

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 Cabello at al. presented 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 type of glyph we discuss are nested disks. Formally, every glyph G^i is given by a set of disks $G_1^i, ..., G_k^i$ such that $G_{j+1}^i \subset G_j^i$ for all $j \in \{1, ..., k-1\}$. Often times figures using nested disks also center all the disks at the same point. The second type of glyphs we use are pie charts. A pie chart is formally given by a disk D_i and a set $\{\alpha_1^i, ...\alpha_k^i\}$ of dividing lines which we can encode as angle in $[0,\ 2\pi]$. Usually the dividing lines separate areas of different color representing different fractions of a sum. Finally, we take a new approach and select squares as glyphs to represent data which are especially interesting as they are not axially symmetrical like circles. Though squares are generally described by vertex sets representing corners of a square or a center and a width our approach allows a description similar to pie charts in the sense of optimality.

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 step calculate for all disks their visible perimeter as if they were the bottommost disk, choose the disk D_i with the largest visible perimeter and add it to a stacking order

$\phi : [n] \to D$ which can be represented as array or list computationally. In the next step we look at the set of disks without D_i and repeat the procedure. This is done until we have processed all disks and thus constructed a stacking order ϕ from bottom up.

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 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, GreedyStacking(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 3 returns a stacking order s which maximizes*

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

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 $s \neq s^*$ that 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

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

Clearly, $s'(i) = s \circ \tau(i) = s(j) = s^*(i)$ 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 $s, s', s'', \ldots, s^*$ 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 $k' \notin \{i, \ldots, j\}$, 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 $k' \in [i+1, \ldots, j]$, 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

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

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, the crux is to define meaningful utility functions for 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 derived for a glyph by only taking the visible percentage of each nested circle into consideration. 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 this happens the algorithm has an optimal solution of value zero. In case of this situation, we follow the heuristic to 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, since it guarantees that at least in this time step the least amount of circles get covered.

Remark. *It is NP-hard to calculate physically realizable drawing where* $\min \Gamma_{min}$ *or* $\min \Gamma_{sum}$ *is maximized since disk glyphs are a special case of nested disks 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 stacking with runtime $O(n^3k^2)$. The runtime could be improved to $O(n^2k \log nk)$ by adapting the data structure introduced in [8].

3.2 Nested disks

For dense regions, centered nested disks lose a lot of information since the small circles can be completely covered up easily. 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, however, now the centers don't have to coincide.

Our approach is straightforward. 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 identify the center of a glyph.

We again use the naive approach for disks from [8] 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 [8] 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. However, the relative distance between all angles must stay the same. Hence, a 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^i_j to the covered perimeter. If the perimeter is completely visible, then this distance is defined by half the circumference or πr_i.

Now we 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 such that the minimum distance of any point in $p \in \bigcup P_i$ to the next covered point on the boundary of d_i is maximized. I.e., we are given Pie-Charts and can like to arrange them by specifying their order and orientation. We would like to find an arrangement of that sort such that all the points on the boundary that separate two pie segments are as far away from the covered intervals as possible.

For the proof we assume a Real-RAM model of computation. A proof assuming only a RAM will require further elaboration on the details of the subsequently provided construction and is beyond the scope of this discussion.

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) such that the minimum distance of any of the distinguished points p_i to the next covered point along the perimeter of its perimeter is greater or equal to k. Or formulated as 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 [8]. We aim to reduce the NP-hardness from $planar$-3-SAT, i.e., instances $\mathcal{I}$ of 3-SAT with the property that the 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 if and only if one of it's literals appears in the clause.

Example:

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

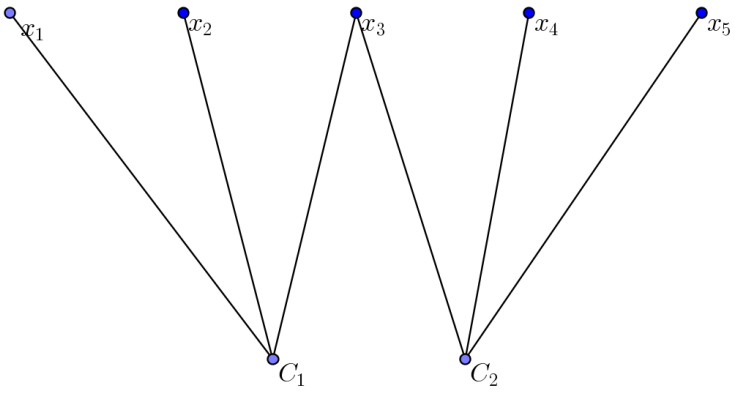


Figure 4: The incidence graph $G(\mathcal{I})$

Given a planar instance $\mathcal{I}$ of 3-SAT we need to construct an instance $\mathcal{D}$ of our pie-charts such that $\mathcal{I}$ is satisfiable if and only if $\mathcal{D}$ 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

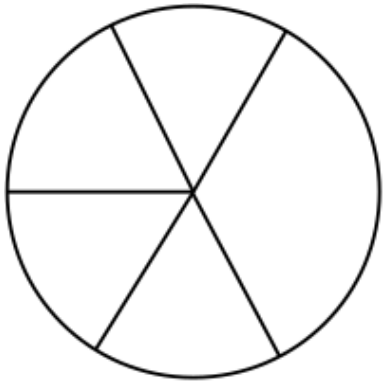


Figure 5: A "neutral" disk used as a connector within a variable ring. The angles are 60 and 120 degrees

We use the three disk types in Figures 5 and 6 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 covered area is 15 degrees, because the segment is covered at 120 degrees and 90. 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 covered area.

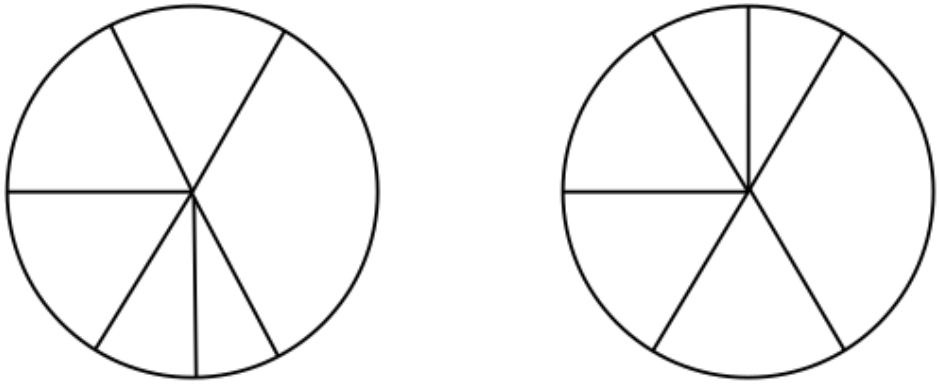


Figure 6: A "true" and "false" disk, similar to the neutral one above, one of the 60 degree segments is cut in half

Figure 7 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.

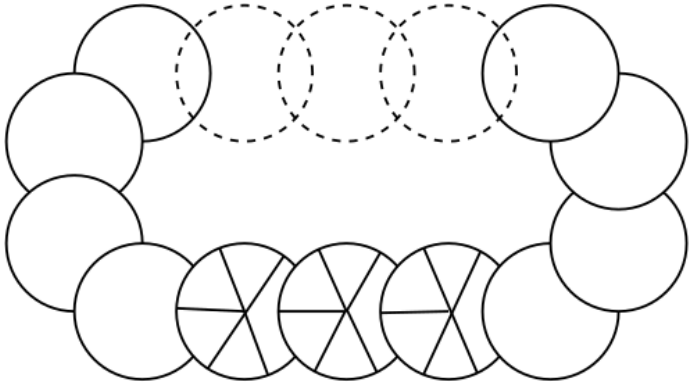


Figure 7: One such gadget G_i for each variable x_i

If a gadget is drawn in such a way that the disks are ascending in a clockwise order, then we will interpret this as the corresponding variable being set to "true" and otherwise as "false".

For each occurrence of the variable x_i in any of the clauses we will substitute a neutral disk in the gadget G_i by either a "true" or "false" disk as depicted in Figure 6. We use a "false" disk if the literal appears negated in the clause.

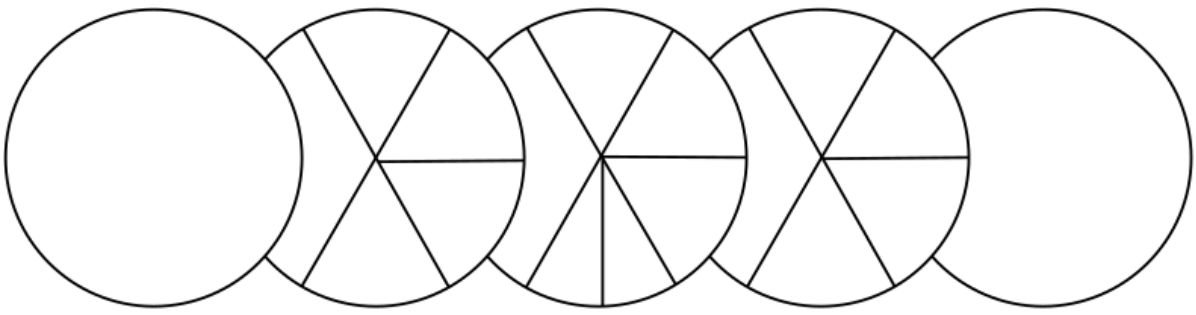


Figure 8: A segment of a "true" gadget with a "false" disk in it

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 the corresponding literal in the clause is false.

Next we will focus on how the clauses will be set up. Later we will connect the gadgets and the clauses, which is why we require the instance to be planar.

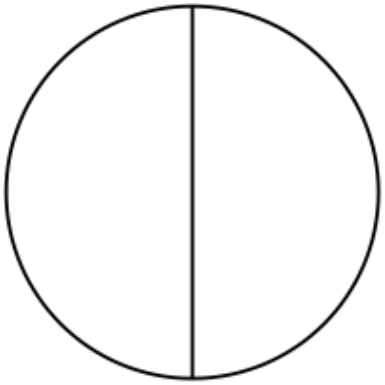


Figure 9: One such disk for each clause

Recall that the instance is satisfiable if and only if each clause is fulfilled. We instantiate one disk as in 9 for each clause. It has two separating lines exactly opposing one another. For each of the three literals in the clause we overlap this disk with one of the disks shown 10 by an angle of 90 degrees as shown in Figure 11.

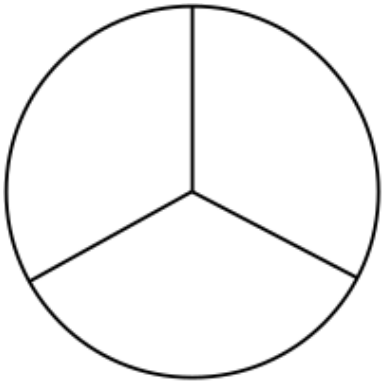


Figure 10: This disk is the endpoint of the edge connecting the gadget to the clause disk

We want to construct the arrangement of pies in such a way that the connecting disk is above the central clause disk if and only if the literal associated with the connecting disk 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, indicating it is "false". If all three connectors go over the clause, then the clause disk cannot have 15 degrees of space between the separating points and the covered area. If, conversely, at least one

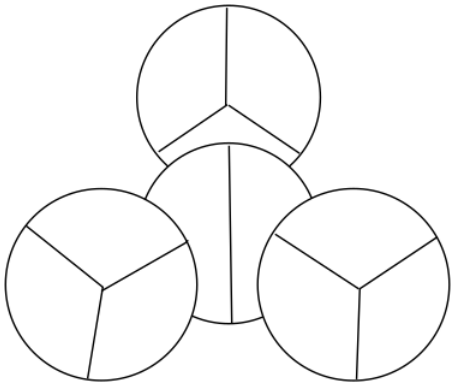


Figure 11: The clause together with the literal connectors

of them goes under, like potentially the top one in Figure 11 then by rotating the clause disk as depicted creates 15 degrees of length between the lines and the covered area.

Now we need to bring it all together. Recall that we have gadgets that can be oriented 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 if and only if the *literal* is false in the clause. We also have connectors at the clause that will act as if "false" if and only if they go over the clause disk, which has to occur if their separating line cannot be arranged as in the top disk of Figure 11.

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.

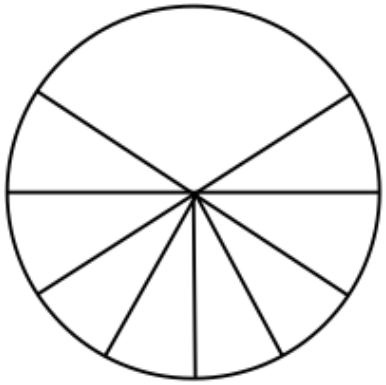


Figure 12: These disks connect the gadgets to the corresponding clauses

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:

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 one and has to go over the clause-disk.

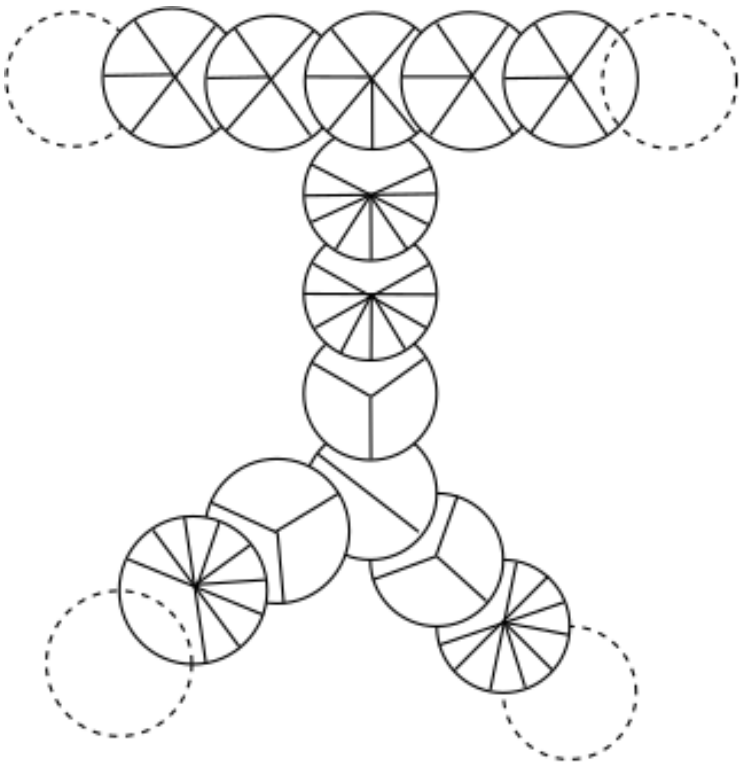


Figure 13: These disks connect the gadgets to the corresponding clauses

Example

We give an example construction for a subset of the 3-SAT instance mentioned in the beginning of this section. The coloring is added to mark important components.

$$\mathcal{I}' = (x_2 \vee \overline{x_3}) \wedge (x_3)$$

The gadget on the right represents variable x_2 and is set to "true" as it is ascending clockwise. The disk representing its occurrence in C_1 is marked in green as it will resolve to "true" and allows the connection to overlap is partially. The gadget on the right represents variable x_3 and is also set to true. It has one negated occurrence in C_1 and one normal occurrence in C_2. The negated occurrence resolves to "false" and the additional delimiter in the red disk is pointing outwards, forcing the first connecting disk to go under it. The dashed disks represent the other connections in a full 3-SAT instance.

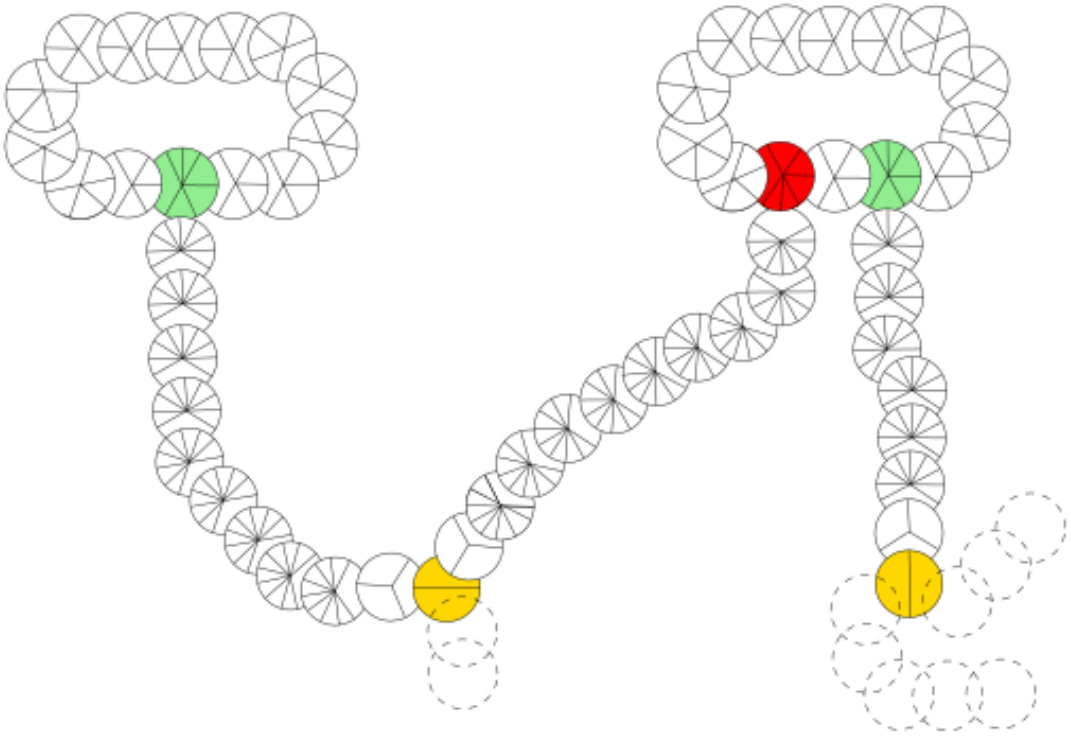


Figure 14: Example construction for the instance $\mathcal{I}'$

Conclusion

The logical properties were already explained when motivating the construction, yet, will be formalized here again. Notice that the disks in the clause and the gadget can create a distance of 15 degrees between cover 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 covered 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 covered 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 covered 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 covered.

Lemma 1. *Given a set of covered 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 $\mathcal{I}$ covers at least one dividing line. Conversely, any rotation locating α_1 in none of the intervals in $\mathcal{I}$ will cover 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 covered. " $\Longleftarrow$ ": Analogously. □

The complement of $\mathcal{I}$ on the circle describes all of the locations of α_1 which do not cover any dividing line. This complement may be empty since there may not be a rotation which does not cover 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 covered 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, the IntervalMidpointAlgorithm returns a rotation which maximizes the minimal radial distance of any of the α_j^i to the covered perimeter.*

Proof. The algorithm rotates a given dividing line in such a way, that it lies exactly in the midpoint of the biggest not covered 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 covered interval maps to the circle's perimeter, it also maximizes the distance to the boundaries of the covered intervals $\{[a_1, b_1], ..., [a_k, b_k]\}$. □

```
1  Input: I = {[a_1, b_1], ...[a_k, b_k]} {α_1, ...α_m})
2    calculate 𝓘 = I ∪ ⋃_{i=2}^{m} ⋃_{j=1}^{k} {[a_j − β_i, b_j − β_i]}
3    let 𝒱 be the set of maximal continuous intervals in [0, 2π] \ 𝓘
4    v = [a_v, b_v] biggest interval in 𝒱
5    r = a_v + (b_v − a_v)/2
6    return r
```

Figure 15: IntervalMidpointAlgorithm

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

If we use a naive implementation for Theorem 5 we get a runtime of $O(n^3k^2)$. As before, the data structure introduced in [8] 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 glyph 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 [8]. 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 step t. Figure 16 illustrates the problem for squares and their visible perimeter. The same applies to squares with subdivisions.

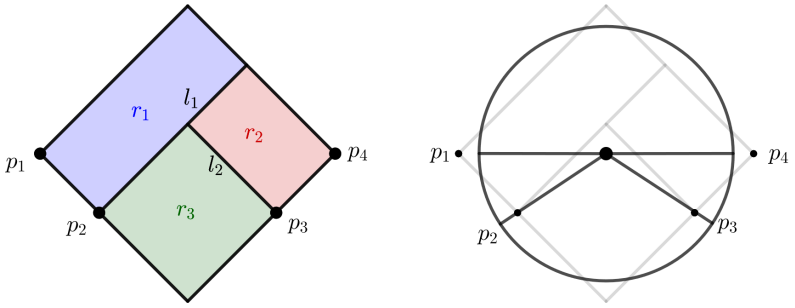
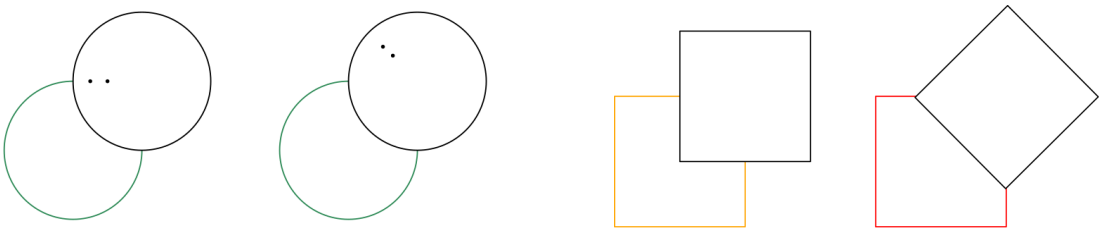


Figure 17: left: square glyph, right: heuristic pie chart



(a) Bottom circle visibility unchanged

(b) Bottom square less visible

Figure 16: Illustration of visibility changes for square rotations

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 visualization 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 a 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 as given by the JHU CSSE COVID-19 Data from [9]. 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 points 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 covered perimeter.

We can get a good solution with the help of pie charts. For every square we construct a 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

We will now compare our stacking algorithms with other stacking methods which are mainly used for visualization of proportional symbol maps. The lab was motivated by the visualization of COVID-19 data. Therefore we will be using the JHU CSSE COVID-19 Data which is given in [9]. The dataset is updated daily and provides data for every day since January 2020. Hence, we have a plethora[1] of data for our experiments.

We will only cherry pick a few data sets as the data quality naturally varies, however, the reader may use our application to look at the results for different dates and inspect interesting data sets in detail. Further, we will always use three features, namely: the unfortunate (1) confirmed deaths [color: black], (2) confirmed recovered [color: green], and (3) overall confirmed cases [color red] of a country.

Since some countries dominate the rest of the countries with regards to the number of cases we scale logarithmically and add 1 to the data to avoid zero radii. To diversify our data sets even more we use three additional parameters in our data preparation: (1) the minimum number of cases to be depicted, (2) a maximum "size" M of the glyphs in dependence of the glyph's type, and (3) a scaling factor S. If we denote the maximal number of confirmed cases in one country by c_{max} then the radius r of the disc of some country with confirmed cases c is given by:

$$r = M \log \frac{cS}{c_{max} + 1}$$

As already mentioned, especially three country (US, Brasil and India) far out-scale other countries, c_{max} is chosen with respect to the next largest case by country at some point in the data. Consequently c_{max} is then set to the 4th highest number of confirmed cases in any given dataset.

The algorithms we will be using for comparison are the Painter algorithm, Left-to-Right and Right-to-Left:

[1] As the data is collected at several points in time under fluctuating political circumstances we have chosen the most trustworthy source. Nevertheless, by simply looking sharply at the data it is obvious that the data is to be taken with a grain of salt.

- The Painter algorithm stacks the glyphs from large to small with respect to the largest circle of the glyph.
- Left-to-Right inserts the Glyphs with respect to increasing x value of the glyph's center.
- Right-to-Left inserts the Glyphs with respect to decreasing x value of the glyph's center.

If we use an algorithm to get optimal rotations we will compare our stacking to the other algorithms with random rotation and with the use of our rotation algorithm. Since most of the glyphs are inherently different we will be evaluating them independently and make an objective comparison at the end.

Nested Disks

algorithm	covered	minVis (rel)	minVis (abs)	min one glyph	average rel vis	absolute perc
random	44	0.011 (0)	0.995 (0)	0	0.658	0.677
LeftToRight	42	0.053 (0)	2.189 (0)	0	0.641	0.678
RightToLeft	43	0.053 (0)	0.995 (0)	0	0.656	0.693
Painter	16	0.064 (0)	6.283 (0)	34.991	0.761	0.718
MinMinStacking (abs)	16	0.075 (0)	2.189 (0)	44.467	0.757	0.724
MinMinStacking (rel)	18	0.11 (0)	2.189 (0)	37.327	0.748	0.725
MinSumStacking (abs)	18	0.111 (0)	3.974 (0)	44.467	0.75	0.721
MinSumStacking (rel)	18	0.111 (0)	2.189 (0)	37.327	0.744	0.723

Table 1: centered nested disks
date: 02.08.2020 , parameters: $M = 50$, $S = 500$ and minimum number of cases $= 5000$

For the nested disks we will be using the following values for our comparison of the algorithms:

- *covered* counts the number of subcircles which are completely covered.
- *minVis (rel)* is given by the minimal relative visible perimeter of any of the subcircles (if some are completely covered its indicated by a 0 in brackets and the first value not equal to zero is used).
- *minVis (abs)* is given by the minimal absolute visible perimeter of any of the subcircles (if some are completely covered its indicated by a 0 in brackets and the first value not equal to zero is used).
- *min one glyph* is given by the minimal value of a glyph, where the value of a glyph is given by the sum over the visible perimeter of all its subcircles.

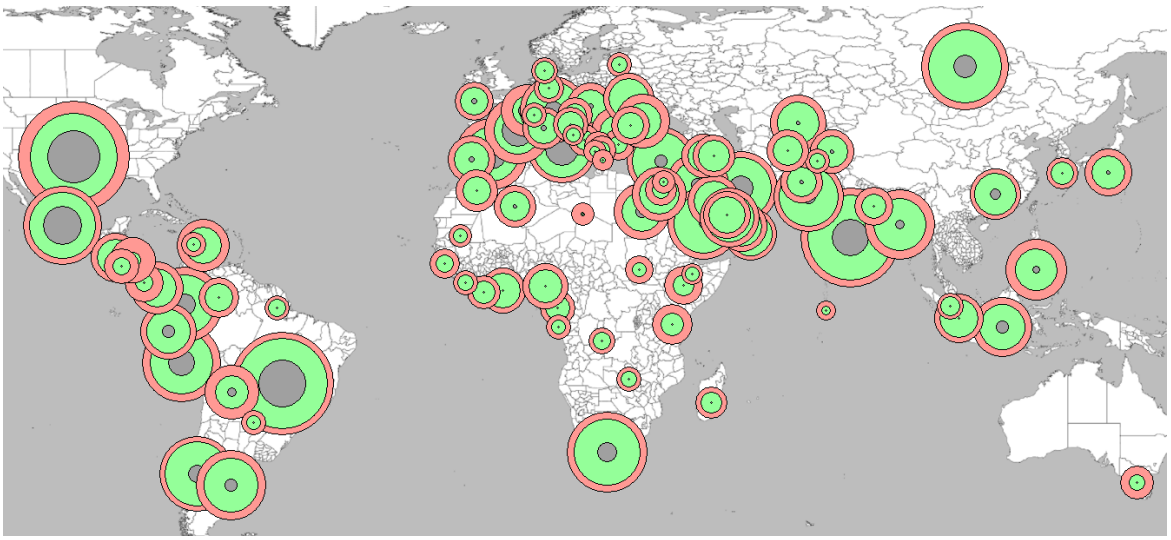


Figure 18: MinMinStacking (Abs) for centered nested disks
date: 02.08.2020 , parameters: $M = 50$, $S = 500$ and minimum number of cases $= 5000$

- *average rel vis* is the average over the visible relative perimeter of all subcircles.
- *absolute perc* is the percentage of visible perimeter of all subcircles.

Our algorithm is designed to maximize the value found in the second column and our theorem about the greedy stacking technique implies its result is optimal. However, in this case the optimum is 0, meaning there is no stacking where no circle is covered. Thus the algorithm applies a heuristic to decide which of the remaining disks to put on the stacking. We see that it minimizes the total number of covered circles, but so does the painters algorithm. Furthermore, the painters algorithm is only marginally beat by our algorithm for relative visibility and much better for absolute visibility.

algorithm	covered	minVis (rel)	minVis (abs)	min one Glyph	average rel vis	absolute perc
random	21	0.001 (0)	0.589 (0)	0	0.765	0.714
LeftToRight	12	0.15 (0)	2.743 (0)	0	0.775	0.725
RightToLeft	13	0.106 (0)	2.89 (0)	0	0.783	0.735
Painter	0	0.093	6.283	47.758	0.857	0.759
our Stacking	0	0.373	6.283	75.034	0.859	0.77

Table 2: nested disks (not centered)
date: 02.08.2020 , parameters: $M = 50$, $S = 500$ and minimum number of cases $= 5000$

In the algorithm for arbitrary nested disks, we stack the disks greedly again, but only care for the local utility of the biggest disk at first. Then, since the disks do not have to be centered, we nest the inner disks such that their boundaries touch in a single point. This point is the midpoint of the longest consecutively visible interval of the outer disk's boundary. We can already note a much better performance. Note that the

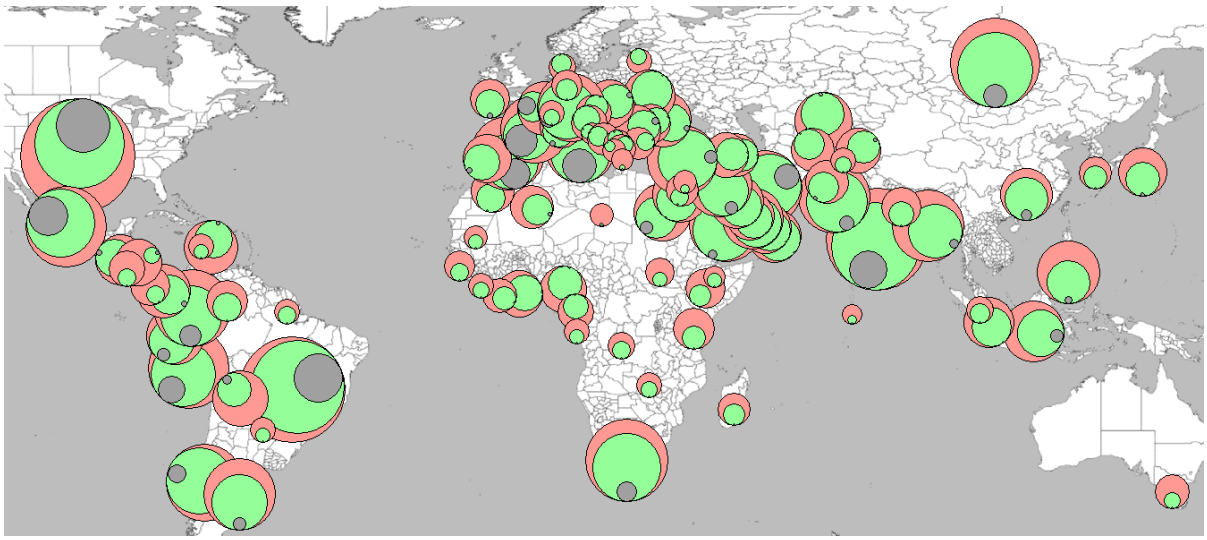


Figure 19: our algorithm for nested disks (not centered)
date: 02.08.2020 , parameters: $M = 50$, $S = 500$ and minimum number of cases $= 5000$

dataset is the same as in the previous example. The increased flexibility of the glyphs makes it possible to attain a utility greater than 0 and allows our algorithm to really harness its potential.

Both our algorithm and the painters algorithm (also using the subroutine described above for the inner disks) arrange all disks such that they are visible. This is already far better than any of the more naive comparison algorithms. Our algorithm then strongly outperforms the painters algorithm on relative visibility and ties in the absolute. This is interesting since, in this case, our algorithm is not guaranteed to return an optimal stacking because our subroutine not necessarily maximizes the local utility of the non-centered glyph.

If we compare the nested centered disks and the arbitrary nested disks we can also see that secondary utilities have increased by up to 10% and the number of covered circles is guaranteed to be be 0. The trade off is just a visual representation which is unusual for the user but does not lose any information.

Pie Charts

For the pie Charts we will be using the following values for our comparison of the algorithms:

- *covered* counts the number of dividing lines which are completely covered.
- *minDist* is given by the minimal value of any pie chart, where the minimal value is given by the radial distance (in $[0, 2\pi]$) to the covered perimeter of the dividing line which is closest to the covered perimeter (if some are completely covered its indicated by a 0 in brackets and the first value not equal to zero is used).

algorithm	covered	minDist (rel)	minDistAvg (rel)	minDistAvg (abs)
Painter+random	80	0.0 (0)	1.01	24.266
random+heuristic	29	0.0 (0)	1.587	42.946
RightToLeft	18	0.017 (0)	1.621	43.407
Painter+ heuristic	5	0.022 (0)	1.706	41.719
our Stacking	0	0.271	1.765	44.452

Table 3: pie charts
date: 22.08.2020 , parameters: $M = 50$, $S = 500$ and minimum number of cases $= 5000$

- *minDistAvg* is given by the average of the values of all of the pie charts.
- *minDistAvg (abs)* is given by the average if we look at the absolute values, i.e radius times angle.

Our pie chart stacking is guaranteed to be optimal by the greedy stacking theorem. Only our algorithm manages to make all separation points visible for each pie glyph. The painters algorithm (using the optimal rotations subroutine) still outperforms more naive approaches having only 6 lines covered. Our algorithm is also better in the secondary utilities although the other algorithms may ”abuse“ the fact that they cover a few of the lines.

The date for this dataset was chosen, such that the optimum is not 0, i.e., it is possible to make all pie separation points visible. Our algorithm also performs well on other instances as can be verified in the application.

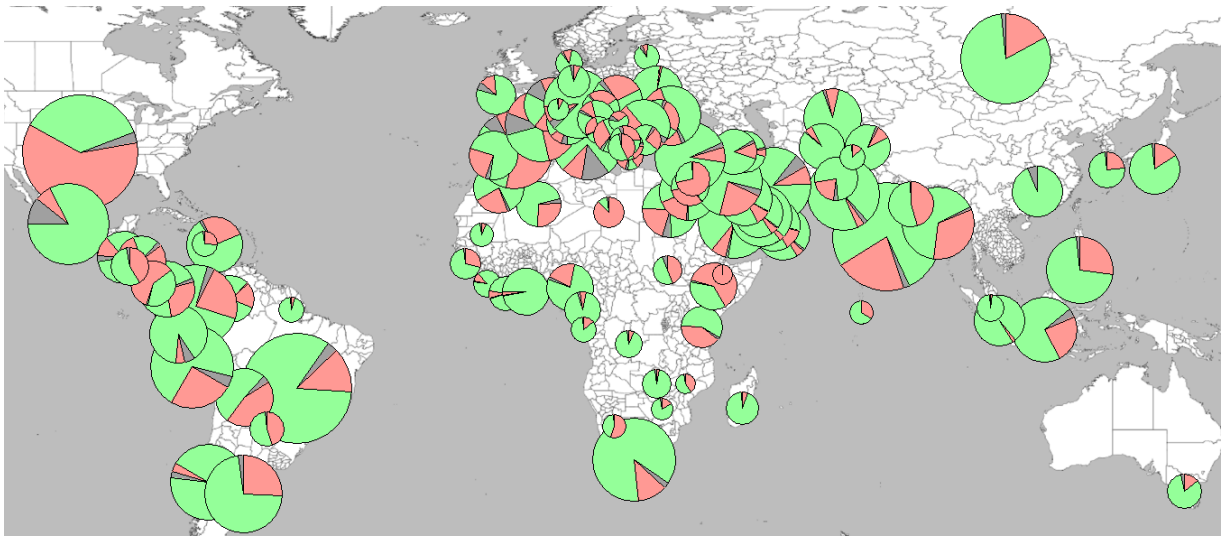


Figure 20: pie Charts
date: 22.08.2020 , parameters: $M = 50$, $S = 500$ and minimum number of cases $= 5000$

algorithm	covered	minDist
random Stacking+random rotations	87	0.235 (0)
Painter+random rotations	41	0.58 (0)
random Stacking+heuristic rotations	56	0.027 (0)
Painter+heuristic	19	0.052 (0)
our Stacking	13	0.052 (0)

Table 4: squares
date: 22.08.2020 , $M = 50$, $S = 500$ and minimum number of cases $= 5000$

Squares

For the squares we will be using the following values for our comparison of the algorithms:

- *covered* counts the number of special points $(p_1, ..., p_4)$ which are completely covered.
- *minDist* is given by the minimal value of any square, where the minimal value is given by the distance on the edge of the square to the covered perimeter of the special point which is closest to the covered perimeter (if some are completely covered its indicated by a 0 in brackets and the first value not equal to zero is used).

We can see that our heuristic of transforming the square mosaic into a pie chart worked well. Our algorithm produces by far the least number of covered separation points. However, it is not guaranteed to be optimal and it does not beat even the naive random approaches on a secondary performance indicator, but the indicator is not very significant since the actual value should be 0.

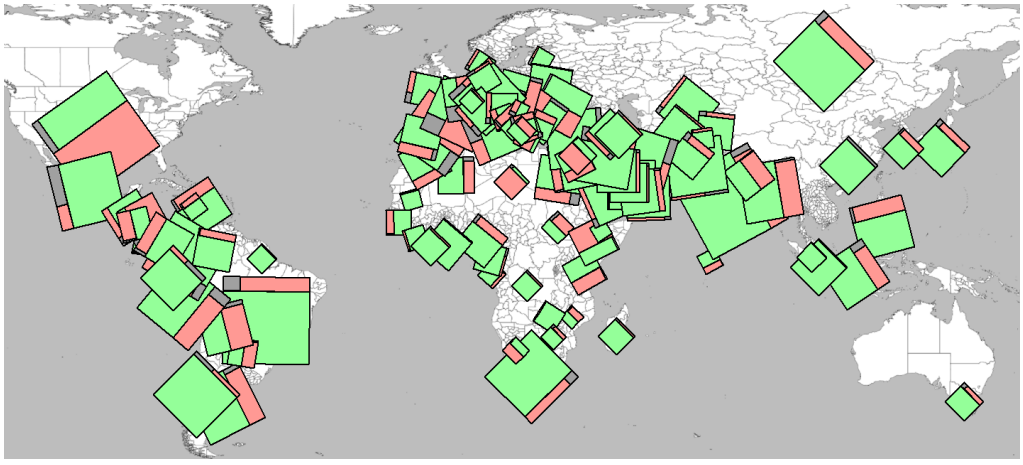


Figure 21: squares
date: 02.08.2020 , parameters: $M = 50$, $S = 500$ and minimum number of cases $= 5000$

5 Outlook & Conclusion

In this work four types of glyphs were compared to each other using a subset of physically realizable drawings called stacking order algorithms. The glyph types were: (1) Centered nested disks, (2) nested disks, (3) Pie charts, and (4) Squares. We introduced several new types of glyphs compared to the mainstream representation: These are (1) the degree of freedom to freely shift inner disks in the nested disks, (2) pie charts with movable proportions and (3) squares. Further, we have shown that 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 as additional degree of freedom, such that the minimum distance of any of the distinguished points p_i to the next covered point along the perimeter of its perimeter is greater or equal to k. In a very similar manner squares as glyphs can be treated in proportional symbolic maps in our approach.

The pie-chart approach enables an utility improvement for the squares as shown in Section 3. Overall runtime is similar and in the order of $O(n^3)$, however, can be brought down to $O(n^2 \log n)$ with a good choice of data structures similar to the approach in [8]. Nevertheless, some heuristics are necessary. Glyphs are a trade-off between content and comparability. At the bottom line it can be safely said that a greedy approach as approximation to the optimal solution is a good choice once more. We found in accordance with [1] that utility is difficult to impossible to be compared among glyphs of different types. As shown in the experiments, disks can be compared to each other as can be pies and squares with the given approach.

Note that different glyphs have a focus on different aspects of the data. The nested disks allow a comparison between sub-disks of different glyphs, e.g., we can compare the number of deceased citizens of the US to the number of confirmed cases in Germany. On the other hand the comparison of the different features in one glyph is difficult, e.g., comparing the ratio of deceased and recovered for any glyph is difficult. For pie charts the exact opposite is the case, as it is easy to make a ratio comparison for one glyph but it's difficult to compare sub-disks of different glyphs. Assuming linear scaling the square approach should be the best of both worlds, as they depict ratios for a single glyph nicely and the sub-rectangles are easily compared among different glyphs. The downside of the squares are a seemingly chaotic visualisation and a far more difficult optimization problem since a simple greedy approach does not work. Even the difficulty of calculating an optimal stacking with optimal rotations needs further investigation.

Furthermore, the ratio between covered points and overall points/lines in the square/pie approach can be seen as a discrete version of the relative visibility which we used for the nested disks. If we go one step further we could arbitrarily add new important points. If we then take the limit, the value of the ratio should approach the relative visibility, which may allow a better comparison. However, this needs to be explored in further work.

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 diskrete geometry: an application to diskrete 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.

[9] Dong E, Du H, Gardner L. An interactive web-based dashboard to track COVID-19 in real time. Lancet Inf Dis. 20(5):533-534. doi: 10.1016/S1473-3099(20)30120-1, `https://github.com/CSSEGISandData/COVID-19`