

Selecting the Aspect Ratio of a Scatter Plot Based on Its Delaunay Triangulation

Martin Fink, Jan-Henrik Haunert, Joachim Spoerhase, and Alexander Wolff

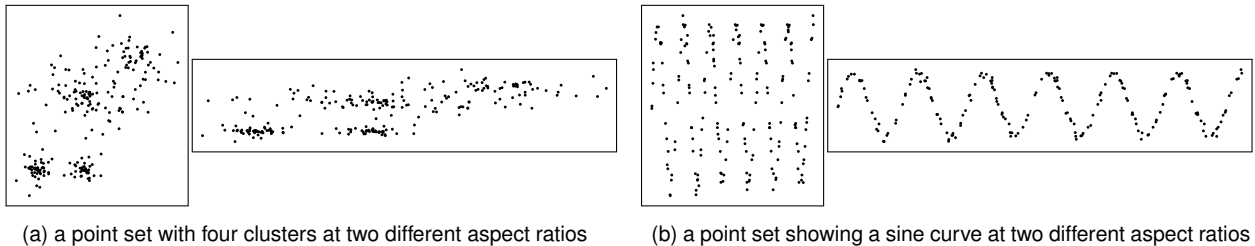


Fig. 1. Choosing the aspect-ratio of a scatter plot has a considerable influence on the readers' impression and their ability to recognize clusters and trends of point data.

Abstract—Scatter plots are diagrams that visualize two-dimensional data as sets of points in the plane. They allow users to detect correlations and clusters in the data. Whether or not a user can accomplish these tasks highly depends on the *aspect ratio* selected for the plot, i.e., the ratio between the horizontal and the vertical extent of the diagram. We argue that an aspect ratio is good if the Delaunay triangulation of the scatter plot at this aspect ratio has some nice geometric property, e.g., a large minimum angle or a small total edge length. More precisely, we consider the following optimization problem. Given a set Q of points in the plane, find a scale factor s such that scaling the x-coordinates of the points in Q by s and the y-coordinates by $1/s$ yields a point set $P(s)$ that optimizes a property of the Delaunay triangulation of $P(s)$, over all choices of s . We present an algorithm that solves this problem efficiently and demonstrate its usefulness on real-world instances. Moreover, we discuss an empirical test in which we asked 64 participants to choose the aspect ratios of 18 scatter plots. We tested six different quality measures that our algorithm can optimize. In conclusion, minimizing the total edge length and minimizing what we call the “uncompactness” of the triangles of the Delaunay triangulation yielded the aspect ratios that were most similar to those chosen by the participants in the test.

Index Terms—Scatter plot, aspect ratio, Delaunay triangulation

1 INTRODUCTION

Scatter plots are widely applied in scientific visualization to find clusters, patterns, and trends in empirical bivariate data. Each data element is visualized as a point in the plane, for example, by using a dot symbol. Additionally, scatter plots often show trend lines or contour lines. According to a survey by Friendly [8], scatter plots appeared in the scientific literature later than pie charts, line graphs, and bar charts, which are generally attributed to William Playfair (1759–1823). Probably, scatter plots have not been used before 1833, when Herschel [12] investigated the relationship between the magnitudes and the spectral classes of stars. Today, however, scatter plots are very common. Tufte [26] notes that “the relational graphic—in its barest form, the scatterplot and its variants—is the greatest of all graphical designs.”

The two dimensions displayed in a scatter plot are usually of different units—consider, for example, a scatter plot that shows the relationship between pairwise measurements of temperature (in degree Celsius) and air pressure (in hPa). Often, each of the two units of the input data is simply mapped to one geometric unit in the drawing.

This approach is inappropriate, however, since a very different visualization would be obtained if the input data was given in different units (for example, in Fahrenheit and Bar). At the worst, this may affect the conclusions that researchers draw about the presence of trends or clusters in their data. Therefore, the mapping between the units of the input data and the coordinates of the drawing needs to be chosen carefully. In this paper, we require that this mapping is a *scaling*. We do not address, for example, scatter plots with a logarithmic scale. We also assume that the area of the plot is prescribed, thus we only have to choose the *aspect ratio* of the drawing, that is, the ratio between its horizontal and its vertical extent.

The problem of finding a good aspect ratio for a scatter plot is closely related to the problem of finding a good aspect ratio for a line chart (that is, the plot of a function), which has been intensively discussed in the literature on information visualization [2, 3, 11, 23, 24]. The existing methods, which we review in Sect. 2 in more detail, mainly rely on properties of the line segments displayed in the diagram. Therefore, they cannot easily be generalized to scatter plots that display points only. A way to overcome this problem, which was proposed by Cleveland et al. [3], is to apply a line-segment-based method to “virtual line segments”, that is, line segments that humans may perceive in a scatter plot though they do not physically exist [22]. Cleveland et al. suggested applying their method to the segments of a (virtual) polyline connecting all data points or the segments of a regression (poly-)line that reflects a global trend. While this approach produces satisfactory results if the data contains a trend, Talbot et al. [23] showed that the trend-line-based approach is inappropriate in the case of two variables that do not have a functional relationship. In this case, it is still interesting to visualize the data, for example, to detect clusters. Therefore, Talbot et al. used contour lines that they computed for the given points with a kernel density estimator (KDE) [19], which is implemented in the statistics package R [16]. Then, they selected the aspect ratio with a line-segment-based method that minimizes the

- Martin Fink is with Lehrstuhl I, Institut für Informatik, Universität Würzburg. E-mail: martin.a.fink@uni-wuerzburg.de.
- Jan-Henrik Haunert is with Lehrstuhl I, Institut für Informatik, Universität Würzburg. E-mail: jan.haunert@uni-wuerzburg.de.
- Joachim Spoerhase is with Lehrstuhl I, Institut für Informatik, Universität Würzburg. E-mail: joachim.spoerhase@uni-wuerzburg.de.
- Alexander Wolff is with Lehrstuhl I, Institut für Informatik, Universität Würzburg. E-mail: alexander.wolff@uni-wuerzburg.de.

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 4 October 2013.

For information on obtaining reprints of this article, please send e-mail to: ivcg@computer.org.

total length of the contour lines. Talbot et al. did, however, not consider that the KDE result heavily depends on the scale between the two dimensions of the input data.

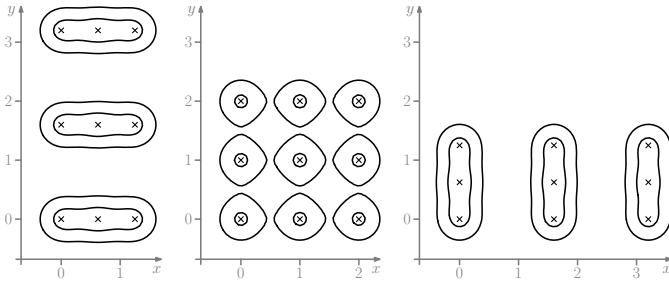


Fig. 2. Scaling a point set (crosses) changes the result of a kernel density estimator and the contour lines derived from it.

Figure 2 illustrates our concern about the contour-line-based method. Suppose that we are given a set of nine data elements that constitute a regular 3×3 grid, see Fig. 2 (center). Clearly, we would say that the data is not clustered. Accordingly, R’s `kde2d` method yields a separate peak for each data element. If we now change the ratio between the vertical and the horizontal spacing of the grid (to simulate a change of the units of the input data), the kernel density estimator groups the points into rows (Fig. 2 (left)) or columns (Fig. 2 (right)). Similarly, since humans tend to group points by proximity [28], a human may incorrectly conclude that the data is partitioned into three clusters. Therefore, we should rescale the axes of the diagrams such that we obtain the regular grid in Fig. 2 (center). This solution will not be found, however, by minimizing the total length of the contour lines in Fig. 2 (left) or (right). In fact, since Talbot et al. [23] showed that their method tends to transform ellipses to circles, the point sets in Fig. 2 (left) and (right) would be scaled in the wrong direction: for each of the clusters enclosed by a set of ellipse-like ovals, the three contained points would move even further together.

Our method for choosing the aspect ratio of a scatter plot reuses the idea of virtual line segments suggested by Cleveland et al. [3]. Other than the method of Cleveland et al. and the contour-line-based method of Talbot et al., however, our method is *not* based on line segments that were computed before optimizing the aspect ratio. Instead, we define based on the output diagram visible to the user (and thus based on the aspect ratio chosen) whether or not two points are linked via a virtual edge. We argue that the aspect ratio has appropriately been set if the virtual edges have nice properties. (Like Talbot et al. [23], we consider minimizing the total edge length, but we also tested other criteria and we will discuss them.) In contrast to the method that optimizes the aspect ratio based on precomputed contour lines, our method ensures that the result is independent of the units of the input data.

To deal with two variables without a functional relationship, we define a virtual line segment for each edge of the Delaunay triangulation $D(P)$ of the set P of points displayed. This defines a meaningful (usually termed the *natural*) neighborhood for P : if for two points $u, v \in P$ there exists a point $w \in \mathbb{R}^2$ such that both u and v are nearest neighbors of w in P , then there exists an edge $\{u, v\}$ in $D(P)$.

The Delaunay triangulation not only has a conceptually meaningful definition, but has also been shown empirically to explain (to some degree) human perceptual grouping. For example, Dry et al. [5] report on an experiment in which humans were shown constellations of stars and asked to display structures in the constellations by connecting pairs of stars with edges. As an average result, 98% of the edges drawn by an individual were edges of the Delaunay triangulation of the stars. On the other hand, on average, an individual did not draw 58% of the Delaunay edges. Therefore, to automate perceptual grouping, the Delaunay triangulation is often reduced to a subgraph, for example, to the relative neighborhood graph [25] or the Gabriel graph [9]. In this paper, we restrict our discussion to the Delaunay triangulation, but we believe that our approach can be generalized to also find an aspect ratio that optimizes favorable properties of, say, the Gabriel graph.

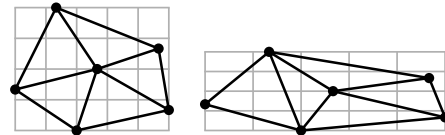


Fig. 3. Scaling a point set may change its Delaunay triangulation.

We now face the problem of choosing the aspect ratio of a scatter plot such that the displayed points have a nice Delaunay triangulation, for example, a Delaunay triangulation with minimum total edge length. This is in fact a challenging algorithmic problem, since—just as the contour lines for a set of points—the Delaunay triangulation may change if we change the aspect ratio, see Fig. 3. To solve the problem, we first consider a naive algorithm that discretizes the interval of possible aspect ratios into a finite but large set S of possible values. Then, we explicitly test each of the $|S|$ values in S . That is, for each $s \in S$, we apply the aspect ratio to the diagram, triangulate the points, and evaluate the total edge length of the triangulation. Finally, we return the aspect ratio that was evaluated best.

The naive algorithm would clearly work, but computing the Delaunay triangulation $|S|$ times from scratch would be extremely inefficient. (Since computing the Delaunay triangulation of a set of n points requires $\Theta(n \log n)$ time, the runtime of the naive algorithm would be in $\Theta(|S|n \log n)$.) To do better, we present an algorithm that traverses the interval of possible aspect ratios while efficiently maintaining the Delaunay triangulation. This algorithm allows us to move from one aspect ratio to the next larger (or smaller) aspect ratio where the Delaunay triangulation changes in $\Theta(\log n)$ time.

Our paper is structured as follows. First, we give an overview over existing methods for aspect-ratio selection for line charts. Next, we precisely define our problem, discuss first ideas, and introduce our quality measures for the aspect ratio of a scatter plot. Then, we present our algorithms for optimizing these quality measures. Finally, we describe our experimental evaluation of the different measures including a user study we performed.

2 EXISTING METHODS FOR ASPECT RATIO SELECTION

The problem of choosing the aspect ratio of a line chart was first discussed by Cleveland et al. [3]. The authors observed that the orientation resolution of two line segments s_1 and s_2 with positive slopes (that is, the absolute difference of their orientation angles) is maximal if the mean orientation of s_1 and s_2 is 45° . In order to allow humans to recognize orientation changes of a line, Cleveland et al. then suggested choosing the aspect ratio such that the median absolute slope of all line segments is 1. This method is usually termed *banking to 45°* and is perhaps the most widely applied automatic method for aspect ratio selection, not least because it is implemented in the statistical software R. Later, Cleveland [2] suggested a weighted version of the method to avoid that redundant line vertices (which would imply an increase in the number of line segments) have an influence on the aspect ratio. In this new version, the length-weighted mean of the absolute orientations of the line segments is set to 45° . Two additional methods for aspect ratio selection have been introduced and tested by Heer and Agrawala [11]. Their first method maximizes the sum of the squared orientation differences over all pairs of line segments; their second method computes the aspect ratio such that the length-weighted mean of the absolute slopes of the line segments equals 1. The most significant innovation of the authors, however, is a multi-scale approach that allows the visualization of a line to be optimized for different levels of granularity. This method finds trends at multiple frequency scales and then generates multiple line charts, of which each optimizes the aspect ratio to visualize the trend at one of the scales.

Talbot et al. [23] suggested choosing the aspect ratio of a line chart by minimizing the total arc length of the line. As the authors show, this method has a close similarity to the method of Heer and Agrawala [11] that maximizes the sum of squared orientation differences (if only pairs of consecutive line segments instead of all pairs of line segments

are considered). In contrast, the method of Talbot et al. is parameterization invariant—redundant line vertices do not influence the result—and faster to compute since only $\Theta(n)$ instead of $\Theta(n^2)$ pairs of line segments need to be considered. The most recent contribution to the topic [24], also by Talbot et al., is an empirical study on how the aspect ratio of a line chart influences humans in estimating the slope of a line. The authors conclude that banking to 45° degrees is not necessarily the best choice and that “substantial future work remains to flesh out a full theory of aspect ratio selection”. With our paper we aim to give fresh impetus to this discussion. We choose a computational-geometry approach that offers new possibilities of optimizing various favorable properties of a scatter plot.

3 PROBLEM STATEMENT AND QUALITY MEASURES

Given a set $Q = \{q_1, q_2, \dots, q_n\} \subset \mathbb{R}^2$ of points, we search for a scale factor $s \in \mathbb{R}^+$ that defines the set $P(s)$ of displayed points, that is, the resulting scatter plot. We denote the coordinates of each point $q_i \in Q$ by x_i and y_i and require that the scatter plot $P(s)$ contains the point $p_i := (s \cdot x_i, y_i/s)$. This ensures that the bounding box of Q and the bounding box of $P(s)$ have the same area. By choosing the scale factor, also the aspect ratio becomes determined. We write P instead of $P(s)$ if the choice of s is clear from the context.

To choose a good scale factor, we define various criteria. Basically, we measure the quality of a scatter plot with a function $f: \mathcal{S} \rightarrow \mathbb{R}$, where \mathcal{S} is the set of all possible scatter plots for the given point set. Then, we search for a scale factor whose corresponding scatter plot maximizes f . Before discussing possible settings of f , however, we introduce two simple methods that are not based on a quality measure.

Simple ideas for choosing an aspect ratio. A first approach would be to compute the bounding box of Q and to scale it to a pre-scribed drawing area, for example a square. With this method, however, at most four points (those with extremal x- or y-coordinates) define the scale factor, while all other points do not influence the result. Therefore, it is left to chance whether the scatter plot allows clusters or trends to be detected. This shows the inappropriateness of the method.

A second, more promising idea is to consider the empirical standard deviations σ_x and σ_y of the x- and y-coordinates of the displayed points. It is somehow reasonable to require $\sigma_x = \sigma_y$ since the variations in both variables may be assumed to be equally important and thus their standard deviations should allocate the same geometric unit. This requirement can be fulfilled by setting

$$s = \sqrt[4]{\frac{\sum_{i=1}^n (y_i - \bar{y})^2}{\sum_{i=1}^n (x_i - \bar{x})^2}} \quad \text{with} \quad \bar{x} = \frac{1}{n} \sum_{i=1}^n x_i, \quad \bar{y} = \frac{1}{n} \sum_{i=1}^n y_i. \quad (1)$$

Since this method is conceptually simple and computationally inexpensive, we include it into the discussion of our experiments in Sect. 5. It is also clear, however, that the standard-deviation-based method has its limitations. Consider, for example, the point set in Fig. 1 (right) scattered around the sine function and assume that it represents a signal we observe, that is, the behavior of a variable (plotted on the y-axis) over time (plotted on the x-axis). Clearly, by observing a sufficiently large number of periods of the signal, the standard deviation of the input x-coordinates gets arbitrarily large. On the contrary, the standard deviation of the input y-coordinates stays constant (or only changes marginally because of the non-periodic nature of the noise). Therefore, by setting $\sigma_x = \sigma_y$ the signal will get more compressed in the horizontal direction the longer we observe it. At some point, we would be unable to recognize the sine function. This shows the deficiency of the standard-deviation-based method. Moreover, the example suggests that we need a measure that quantifies the quality of every part of the scatter plot *locally*, for example, to ensure that every period of a sine signal gets nicely scaled. We need a definition of locality, a local measure of quality, and a function that aggregates the local measures into a single *global* score that is to be optimized.

Quality measures. As pointed out in the introduction, experiments have shown [5] a strong connection between human perceptual



Fig. 4. Triangulations of a point set with incircles of the triangles.

grouping in a given point set P and the edges of its Delaunay triangulation $D(P)$. This motivates our study of Delaunay triangulations as the underlying structure for finding a suitable aspect ratio of a scatter plot. To give an intuitive explanation why the Delaunay triangulation yields a meaningful neighborhood, consider the basic example of Figure 4 where the two possible triangulations of a point set are shown. A possible explanation (for alternative explanations see below) why the triangulation to the left seems less natural than the right one is that it contains very small angles between adjacent edges leading to an “uncompact” appearance. In fact, it is well known that among all triangulations of a given point set (with a fixed scale factor) the Delaunay triangulation (which is the triangulation to the right in Figure 4) is the one whose *smallest angle* is maximal [21].

The idea behind our first quality measure is therefore to maximize the smallest angle also over different scale factors. That is, we choose the scale factor such that the smallest angle of $D(P)$ is as large as possible. Indeed, we can find the scale factor that optimizes this criterion. Our experiments, which we will discuss in Sect. 5, showed, however, that this sometimes yields unsatisfactory results.

A problem with maximizing the minimum angle is that an overall good scatter plot is rated poor if its Delaunay triangulation contains only one very small angle, which may dictate the overall outcome. Therefore, we need a measure that better aggregates local properties of the scatter plot. Interestingly, the Delaunay triangulation not only maximizes the minimum angle but also the *mean inradius* of the triangles [14]. (See also Figure 4.) Since this aggregating measure has the potential to be more robust than a max-min measure when applied to an interval of scales, we also consider maximizing the mean inradius of the triangles of $D(P)$.

While our motivation for the first two measures was mainly driven by properties of the Delaunay triangulation, we also consider measures that have earlier been proposed for selecting the aspect ratio of a line chart. In particular, inspired by the idea of Talbot et al. [23], who defined the aspect ratio of a line chart by minimizing the length of the displayed line, we consider minimizing the *overall length* of the edges of $D(P)$. Similarly, inspired by an idea of Heer and Agrawala [11], who proposed minimizing the square sum of orientation differences of line segments, we consider minimizing the *square sum of the angles* of the triangles of $D(P)$.

Furthermore, we suggest maximizing the *mean compactness* of the triangles of $D(P)$. We define compactness as MacEachren [15] did for the analysis of geographic shapes, that is, we define the compactness of a triangle Δ with perimeter $c(\Delta)$ and area $A(\Delta)$ as $\sqrt{A(\Delta)}/c(\Delta)$. Finally, instead of maximizing the average compactness, we consider minimizing the *mean uncompactness* of the triangles of $D(P)$, where the uncompactness of a triangle Δ is defined as $c(\Delta)/\sqrt{A(\Delta)}$ [15]. We remark that our uncompactness measure may be viewed as a weighted variant of the above length measure. In fact, both measures behaved similarly in our experiments; uncompactness was slightly better. The uncompactness criterion weighs each edge by a factor that depends (reciprocally) on the areas of the incident triangles. This increases the impact of edges incident to skinny triangles while lowering the impact if the incident triangles are compact anyway.

Note that, for a fixed scale, all of the above measures prefer the right over the left triangulation in Figure 4.

To wrap up, we propose six optimization criteria, which we group into *angle-based*, *length-based*, and *triangle-based* measures, namely

- (1) maximizing the minimum angle of the triangles of $D(P)$,
- (2) minimizing the square sum of the angles of the triangles of $D(P)$,
- (3) minimizing the total edge length of $D(P)$,
- (4) maximizing the mean inradius of the triangles of $D(P)$,
- (5) maximizing the mean compactness of the triangles of $D(P)$, and
- (6) minimizing the mean uncompactness of the triangles of $D(P)$.

In the next section, we present an exact algorithm for maximizing the minimum angle and an algorithm that approximates an optimal solution for any of the other five criteria.

4 OUR ALGORITHM

Finding the optimum scale factor s can be seen as a process of continuously increasing s starting with $s = 1$. In doing so, the Delaunay triangulation undergoes *topological changes* at certain *event points* which we need to keep track of. We output the scale factor at which our objective function is optimized. Using the same approach we can traverse all scale factors $s < 1$ and, hence, find the global optimum.

Our algorithm consists of two layers. The first layer steps through the discrete set of event points in the order in which they occur in the above described process. The second layer optimizes between the event points. Between two consecutive event points s_i, s_{i+1} the topological structure of the Delaunay triangulation does not change. Each of our optimization measures is a continuous function of $s \in [s_i, s_{i+1}]$. We can compute the scale factor (or an approximation of it) at which this function is maximized within the interval $[s_i, s_{i+1}]$. Doing this for all such intervals allows us to determine the globally optimal scale factor (or an approximation).

Our approach is closely related to the problem of maintaining a Delaunay triangulation for a set of continuously moving points. In fact, the first layer of our algorithm may be seen as a special case of this problem where each point moves along a horizontal line at an individual but constant speed. To see this, map any input point (x_i, y_i) to $(s^2 x_i, y_i)$ instead of mapping it to $(s x_i, y_i/s)$ at scale factor s . Clearly, this does not change the topology of the triangulation. Substituting s^2 by s shows that it is sufficient (at least in the first layer where only the topological structure of the Delaunay triangulation is relevant) to map (x_i, y_i) to $(s x_i, y_i)$. That is, in the first layer we only need to scale x -coordinates by s while leaving the y -coordinates unchanged.

For the general setting of moving points, Roos [17] describes a data structure for maintaining (the topological structure of) a dynamic Delaunay triangulation. His data structure takes $O(\log n)$ time per topological change. His results requires that the movement of the points meets a (weak) technical assumption that holds for many natural scenarios such as movement along parametric polynomial curves.

The question of *how many* topological changes a dynamic Delaunay triangulation can undergo (again under some weak and natural assumptions on the movement) is an important field of research in computational geometry [7]. There is a series of lower or upper bounds to the problem [10, 17, 20, 1, 18]; closing the gap is a major open problem. In a recent breakthrough, Rubinfeld [18] shows that there are at most $O(n^{2+\epsilon})$ topological changes for a large class of movements (including our scenario). This almost matches the currently best quadratic lower bound [20]. We argue that in our case there can be in fact only $O(n^2)$ topological events and that this bound is tight. Therefore, we are able to close the gap in our setting.

The following characterization of a Delaunay triangulation $D(P)$ of a point set $P = \{p_1, \dots, p_n\}$ is crucial for our algorithm; it consists of two equivalent properties:

- (P1) The interior of the circumcircle of any triangle of $D(P)$ does not contain points of P .
- (P2) For any edge $p_i p_j$ of $D(P)$ there exists a circle whose boundary contains p_i and p_j but whose interior does not contain points of P .

The converse also holds. Any triangulation that satisfies (P1) or (P2) is a Delaunay triangulation.

4.1 Maintaining the Delaunay Triangulation Through Scale Space

Now we turn to tracking the Delaunay triangulation through scale space. Given a set P of n points, h of which lie on the convex hull, it is well-known that any triangulation of P contains $2n - 2 - h$ triangles and $3n - 3 - h$ edges. Clearly, the number of points on the convex hull is the same at any scale. Hence, the numbers of triangles and edges do not change either. Therefore, for each triangle (or edge) that disappears from the Delaunay triangulation at some event point s^* , a

new triangle (or edge) is created and vice versa. For the sake of presentation, we assume in what follows that no five points are co-circular at any fixed scale factor.

Consider an event point s^* at which some triangle disappears from the Delaunay triangulation. According to property (P1) there is at least one point p_r that enters the circumcircle $C(p_i, p_j, p_k)$ of the triangle at s^* . More precisely, the interior of $C(p_i, p_j, p_k)$ contains p_l at any $s > s^*$ but not at $s = s^*$, where p_l is on the boundary.

We first characterize the situations at which we have to perform topological changes. To this end, let a D -quadrilateral be the union of two triangles of the Delaunay triangulation that share an edge.

Lemma 1. *If the Delaunay triangulation undergoes a topological change at event point s^* , then there is a D -quadrilateral $p_i p_j p_k p_l$ with diagonal $p_i p_k$ such that p_l enters the circle $C(p_i, p_j, p_k)$ at s^* .*

Proof. There must be a triangle $\Delta p_i p_j p_k$ that disappears at s^* because there is a topological change at s^* . This means that there is a point p_r entering $C(p_i, p_j, p_k)$, w.l.o.g. between p_i and p_k . Let p_l be the vertex of the triangle adjacent to $\Delta p_i p_j p_k$ and incident to edge $p_i p_k$ as depicted in Figure 5. We show that $p_r = p_l$, which completes the proof. Assume to the contrary that $p_r \neq p_l$. The point p_l must lie outside the circle $C(p_i, p_j, p_k)$ since no five points are co-circular. But then the interior of $C(p_i, p_l, p_k)$ contains p_r , which is a contradiction. \square

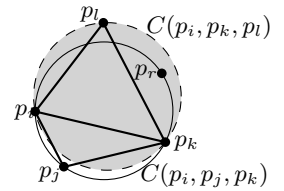


Fig. 5. Proof of Lemma 1.

Consider a point p_l entering the circumcircle of a triangle $\Delta p_i p_j p_k$ at event point s^* as described in Lemma 1. This situation is depicted in Figure 6.

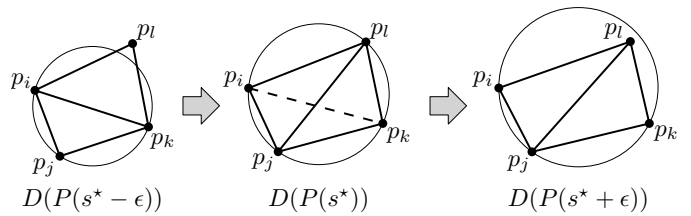


Fig. 6. An elementary topological change: a flip at scale s^* .

Now, if we replace edge $p_i p_k$ with edge $p_j p_l$ at event point s^* we obtain a new triangulation. We call this operation a *flip*. The crucial observation is that the interiors of the circumcircles of the newly formed triangles $\Delta p_i p_j p_l$ and $\Delta p_j p_l p_k$ are empty at scale $s^* + \epsilon$ for some sufficiently small $\epsilon > 0$ because no point other than p_l enters $C(p_i, p_j, p_k)$ at s^* . Therefore, if no further flips are to be performed at s^* , the current Delaunay triangulation is valid at $s^* + \epsilon$ for any sufficiently small $\epsilon > 0$. Also note that the flip of $p_i p_k$ corresponds to the co-circularity of the unique quadrilateral $p_i p_j p_k p_l$ that contains $p_i p_k$.

It can happen that more than one flip has to be performed at s^* . Observe, however, that the four edges of the quadrilateral $p_i p_j p_k p_l$ need not be flipped at s^* because otherwise a fifth point would have to enter its circumcircle (contradicting our assumption). This means in particular that multiple flips that are performed at s^* can be carried out independently of each other.

Our algorithm determines the sequence s_1, \dots, s_m of event points one by one in increasing order starting with $s_1 := 1$. Given an event point s_i and the corresponding Delaunay triangulation $D(P(s_i))$, we need to compute the next event point s_{i+1} , that is, the smallest scale factor larger than s_i at which we have to perform flips. Note that any flip that we have to perform at s_{i+1} corresponds to the co-circularity of the unique quadrilateral of $D(P(s_{i+1}))$ containing the edge flipped (see Figure 6). In other words, for every edge, we have to compute the smallest scale factor larger than s_i at which the corresponding quadrilateral becomes co-circular. For each edge, this event point can be

Algorithm 1: FindAspectRatio(P)

```

 $s = 1$ 
 $(V, E) = D(P)$  // Delaunay Triangulation
 $Q = \text{new PriorityQueue}()$ 
foreach  $e \in E$  do
  Find event point  $t \geq s$  where  $e$  is flipped.
   $Q.\text{Insert}(t, e)$  // args: priority, edge
while  $Q \neq \emptyset$  do
   $ptr = Q.\text{ExtractMin}()$ 
   $t = ptr.\text{event point}$ 
   $e = ptr.\text{edge}$ 
   $f = \text{Flip}(e)$ 
   $E' = \text{set of edges of quadrilateral containing } f$ 
  foreach  $g \in E'$  do
    update event point for  $g$  in  $Q$ 

```

computed in constant time by solving a polynomial equation on s of constant degree (Roos [17] describes more details). The next event point s_{i+1} is then the minimum of the resulting event points.

Algorithm 1 describes how we traverse the sequence s_1, \dots, s_m of event points in increasing order. Initially, we compute the Delaunay triangulation at $s := s_1 = 1$ and set up a priority queue Q that maintains for each edge of the current triangulation the event point at which this edge has to be flipped.

The sequence of event points is computed in the **while** loop. First, the next event point t is computed by extracting from queue Q the next scale factor t at which some edge has to be flipped. We also retrieve the corresponding edge e from Q . Note that t may be equal to the current event point s since more than one edge may be flipped at s .

Whenever an edge e is flipped, the queue Q has to be updated accordingly. This affects the set E' of the four edges of the quadrilateral containing f . For each such edge, its corresponding quadrilateral has changed due to the flip of e , which may require to update the event point at which the edge must be flipped.

Let us now analyze the running time of the algorithm. The initialization step takes $O(n \log n)$ time for computing the Delaunay triangulation and $O(n)$ time for building the priority queue. For each flip that is performed by the **while** loop we spend overall $O(\log n)$ time for extracting the minimum of Q and updating the four edges of the corresponding quadrilateral.

It remains to determine the maximum number of flips performed by the algorithm. Consider the situation depicted in Figure 6 where we flip the edge $p_i p_k$ at event point s^* . Observe that, for every scale factor $s > s^*$, the circle $C(p_i, p_j, p_k)$ contains p_l in its interior. Therefore, every circle with p_i and p_k on its boundary either contains p_j or p_l in its interior. By property (P2) we can conclude that the edge $p_i p_k$ cannot be part of a Delaunay triangulation for any $s > s^*$. Since there are at most $O(n^2)$ potential edges (one for each point pair) and since no edge that has been flipped can be re-inserted into the Delaunay triangulation (as we have just seen), the algorithm performs at most $O(n^2)$ flips. Let us summarize.

Theorem 1. *Algorithm 1 traverses the sequence s_1, \dots, s_m of all event points in increasing order and computes for every $i = 1, \dots, m-1$ the (graph of a) Delaunay triangulation that is valid in the interval $[s_i, s_{i+1}]$. It performs $O(n^2)$ topological changes (flips) each of which requires $O(\log n)$ time. The overall running time is $O(n^2 \log n)$.*

There is a simple input instance showing that our above analysis is asymptotically tight. Let r and s be line segments of slope -1 that lie in the first and third quadrant, respectively. For any positive (even) integer n , we obtain a worst-case instance $P_{wc}(n)$ by picking $n/2$ arbitrary points on r and $n/2$ arbitrary points on s . It is not hard to verify that our algorithm—and any algorithm that maintains the Delaunay triangulation explicitly—performs $\Omega(n^2)$ topological changes.

4.2 Finding a Good Scale Factor

In general, the event points described in the previous section can be scattered quite unevenly over scale space. In order to find a good scale factor it is therefore not sufficient to consider only the event points as potential solutions. In this section, we want to find good solutions for scale factors between event points.

4.2.1 Finding an Approximate Solution

We describe our method for the objective of minimizing the mean uncompactness u_D of the Delaunay triangulation. The method can be applied similarly to the other objective functions listed in Sect. 3.

Fix an interval $[s_i, s_{i+1}]$ of consecutive event points and fix an arbitrarily small error parameter $\varepsilon > 0$. Our goal is to find an $(1 + \varepsilon)$ -approximate solution, that is, a scale factor s_{app} for which $u_D(s_{\text{app}}) \leq (1 + \varepsilon)u_D(s_{\text{opt}})$, where s_{opt} is the globally optimal scale factor.

Let $e = p_1 p_2$ be an edge of the Delaunay triangulation in this interval with end points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, and let $l_e(s) = \sqrt{s^2(x_1 - x_2)^2 + (y_1 - y_2)^2/s^2}$ denote the length of e as a function of $s \in [s_i, s_{i+1}]$. The crucial observation is that this length function behaves smoothly with respect to small changes in the scale factor. More precisely, we have that $l_e((1 + \varepsilon)s) \leq (1 + \varepsilon)l_e(s)$.

Now consider a triangle Δ of the Delaunay triangulation with edges e_1, e_2, e_3 . Its uncompactness is given by $u_\Delta(s) = (l_{e_1}(s) + l_{e_2}(s) + l_{e_3}(s)) / \sqrt{A(\Delta)}$ where $A(\Delta)$ denotes the area of Δ , which is independent of s . The above inequality yields that $u_\Delta((1 + \varepsilon)s) \leq (1 + \varepsilon)u_\Delta(s)$. Recall that $u_D(s)$ denotes the mean uncompactness of $D(P(s))$. Hence, we have that $u_D((1 + \varepsilon)s) \leq (1 + \varepsilon)u_D(s)$ since $u_D(s)$ is the mean of the functions $u_\Delta(s)$ over a fixed set of triangles (due to $s \in [s_i, s_{i+1}]$).

We restrict ourselves to scale factors between 1 and C for some sufficiently large constant C , which is sufficient for practical purposes. Let s_1, \dots, s_m denote the sequence of event points (between 1 and C) and let $s_{m+1} := C$.

Now consider a fixed interval $[s_i, s_{i+1}]$ with $i = 1, \dots, m$. Our algorithm computes $u_D(\cdot)$ for all test values $t_j := s_{i+1}/(1 + \varepsilon)^j$ where $j \in \mathbb{N}$ and $t_j \in [s_i, s_{i+1}]$. Let t_{opt} be the test value at which $u_D(\cdot)$ is minimized, and let s_{opt} be an optimum scale factor in the interval $[s_i, s_{i+1}]$. We claim that $u_D(t_{\text{opt}}) \leq (1 + \varepsilon)u_D(s_{\text{opt}})$, that is, we obtain an $(1 + \varepsilon)$ -approximation for the current interval. To see this, let t_k be the smallest test value such that $t_k \geq s_{\text{opt}}$, which implies that $t_k/(1 + \varepsilon) < s_{\text{opt}} = t_k/(1 + \varepsilon')$ for some $\varepsilon' \leq \varepsilon$. Then we can conclude that $u_D(t_{\text{opt}}) \leq u_D(t_k) = u_D((1 + \varepsilon')s_{\text{opt}}) \leq (1 + \varepsilon')u_D(s_{\text{opt}}) \leq (1 + \varepsilon)u_D(s_{\text{opt}})$ as desired. Finding the test value that minimizes the mean uncompactness over all intervals gives therefore a global $(1 + \varepsilon)$ -approximation.

Now let us analyze the running time of the algorithm. The number of test values in the interval $[s_i, s_{i+1}]$ is the smallest integer j for which $s_{i+1}/(1 + \varepsilon)^j \leq s_i$ holds. That is, we have at most $(\log s_{i+1} - \log s_i) / \log(1 + \varepsilon) + 1$ many test values in this interval. Summing this term over all $O(n^2)$ intervals yields $(\log s_{m+1} - \log s_1) / \log(1 + \varepsilon) + O(n^2)$ test values in total. Since $s_{m+1} = C$ and ε and C are constants and evaluating the objective function requires $O(n)$ time, we obtain the following result.

Theorem 2. *For any fixed $\varepsilon > 0$, we can compute a $(1 + \varepsilon)$ -approximate solution for minimizing the mean uncompactness (given a constant upper bound on the scale factor). The algorithm takes $O(n^3)$ time. Taking ε into account, the running time is $O(n^3 + n/\log(1 + \varepsilon))$.*

4.2.2 Maximizing the Minimum Angle

We now sketch an efficient algorithm for determining the scale factor s that maximizes the smallest angle of the Delaunay triangulation of the scatter plot at scale s .

Recall the continuous process that we described in the previous section where we maintain a dynamic Delaunay triangulation with increasing scale factor s . Each angle β (specified by two line segments) that occurs during this process can be described as a function on the scale factor s . The angle β is not necessarily present over the whole scale space because its defining line segments may not be part of the

Delaunay triangulation all over the scale space. Therefore, the domain of β is a subset of the interval $[1, +\infty]$. As we argued in the previous section, any edge that disappears from the Delaunay triangulation will not reappear at a larger scale factor. Therefore, the domain of β is actually an interval $[l_\beta, r_\beta]$. Let \mathcal{A} be the set of all angles (functions) that appear at some scale factor in the Delaunay triangulation and let $\text{env}(\mathcal{A})$ be the lower envelope of \mathcal{A} . Then determining the scale factor that maximizes the smallest angle of the Delaunay triangulation amounts to determining the maximum of $\text{env}(\mathcal{A})$.

Now consider some angle $\beta \in \mathcal{A}$, and let $p_i p_j$ and $p_j p_k$ be the line segments defining β . Because every triangle has an angle smaller than $\pi/2$, the angle β can only contribute to $\text{env}(\mathcal{A})$ if $\beta(s) \leq \pi/2$ for some $s \in [l_\beta, r_\beta]$. We set up a coordinate system with origin p_j . Then β is strictly larger than $\pi/2$ for any s if p_i and p_k lie in diagonally opposite quadrants. Therefore, we can safely remove from \mathcal{A} all angles whose line segments lie in opposite quadrants.

Under this assumption, it is not hard to verify that any $\beta \in \mathcal{A}$ can be expressed as $\beta(s) = c_1 \pi + \arctan \frac{c_2 s}{c_3 s^2 + 1}$ where $c_1 \in \{0, 1\}$ and $c_2, c_3 \in \mathbb{R}$ are easily computable constants that only depend on the line segments defining β but not on s . Elementary calculations reveal that two functions of the above form can have at most one intersection.

Agarwal and Sharir [20] show that the lower envelope of a set of m partially defined functions, where two functions can intersect at most r times, has complexity (number of distinct curve segments) at most $\lambda_{r+2}(m)$. Moreover, the lower envelope can be constructed algorithmically in $O(\lambda_{r+1}(m) \log m)$ time. Here $\lambda_q(m)$ denotes the Davenport-Schinzel sequence of order q .

To apply these results to our setting let m be the number of angles in \mathcal{A} . Because two functions (angles) in \mathcal{A} can intersect at most once the lower envelope $\text{env}(\mathcal{A})$ has complexity at most $\lambda_3(m)$. It is known that $\lambda_3(m) = O(m\alpha(m))$ [20], where α denotes the extremely slowly growing functional inverse of the Ackermann function.

The lower envelope can be constructed in $O(\lambda_2(m) \log m)$ time. It is known that $\lambda_2(m) = O(m)$ [20]. For each curve segment of the lower envelope, the maximum can be computed in constant time. Because the complexity of the lower envelope is $O(m\alpha(m))$, we can compute the globally optimal scale factor in $O(m \log m)$ time.

Now let us examine how large the number m of angles in \mathcal{A} is in terms of the number n of points. Note that, for any scale factor, the number of angles in the Delaunay triangulation is $O(n)$. Any flip (see Figure 6) that we perform in the scaling process removes four angles from the triangulation and introduces four new angles. Since at most $O(n^2)$ flips are performed, \mathcal{A} contains at most $m = O(n^2)$ angles. Let us summarize.

Theorem 3. *The scale factor that maximizes the smallest angle in the Delaunay triangulation of a set of n points can be computed in $O(n^2 \log n)$ time.*

Alternatively, we propose the following algorithm, which is much easier to implement and fast in practice. The algorithm steps through the set of event point as described in the previous section. It maintains, for each interval of consecutive event points, the collection of angles (functions of s) that are present in that interval. This means that, for any flip that is performed, the collection of angles has to be updated, which takes $O(1)$ additional time (besides the flip) because a constant number of angles is removed or added to the collection. Within each interval, the algorithm traverses the intersection points of curves (angles) on the lower envelope by increasing scale factor s . Given some intersection point on the lower envelope, the next intersection point (in the interval) can be determined in $O(n)$ time by computing the intersection of the currently smallest function with any of the other $O(n)$ functions, which needs $O(1)$ time for each function. To summarize, our algorithm traverses intersection points on the lower envelope and spends $O(n)$ time at each intersection point. Since there are $O(n^2 \alpha(n))$ intersection points, our algorithm takes $O(n^3 \alpha(n))$ time. Our experiments have shown that this algorithm is fast in practice.

5 EXPERIMENTAL RESULTS

In order to analyze the quality of our methods, we implemented our algorithms and tested them on data showing clusters and trends. To evaluate the output of our algorithms, we conducted a user study in which users selected their preferred aspect ratio for the same data sets. We then measured how closely our different methods matched the preferences of users in our study.

5.1 Implementation and Test Instances

We implemented our algorithms in Java. For maximizing the minimum angle, we used the simplified version of our exact algorithm mentioned at the end of Sec. 4.2.2. For the other criteria, we used the approximation algorithm with $\varepsilon = 0.01$.

Table 1 shows results on five test instances: (a) a cluster of points distributed normally around a center; (b) three such clusters next to each other; (c) a collection of four clusters of varying shape; (d) points sampled along a sine function at distances that are distributed normally; (e) the same for a linear trend with varying standard distance.

5.2 Results on Clustered Point Sets

A feature that often occurs in scatter plots are *clusters* of points. It is of course desirable that clusters can be identified easily. Therefore, we tested the optimization criteria on input data containing clusters. We generated clustered point sets by first choosing cluster centers, and then iteratively adding randomly generated points around each center at normally distributed distances with equal standard deviation; see the first two columns in Table 1 for instances with one and three clusters, respectively.

As one simple cluster tends to be symmetric, it should be visualized at an aspect ratio at which the cluster looks like a circular point cloud. In our tests, this worked well for all optimization criteria.

The second column of Table 1 shows point sets with multiple (here: three) clusters placed in a row. Here, the standard deviation and, in particular, the inradius criterion gave bad results, that is, they scaled the point set too much in y-direction. For such instances, the best results were achieved by minimizing squared angles, maximizing compactness, and minimizing uncompactness. The resulting scatter plots separate the clusters well and display each as a nice, almost circular, point cloud.

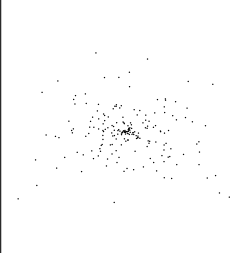

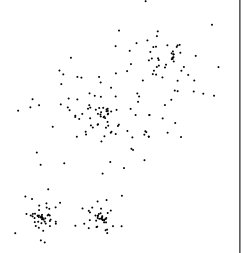
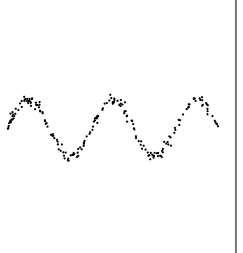
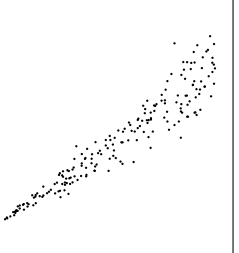
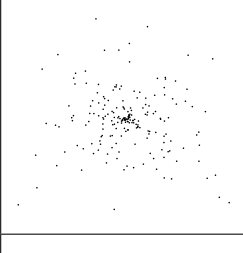
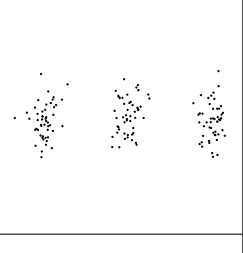
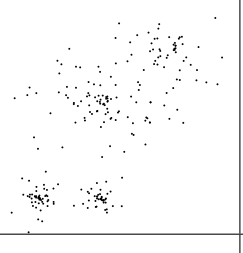
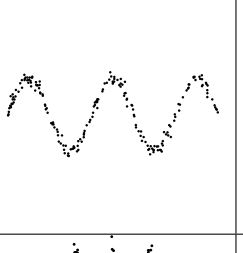
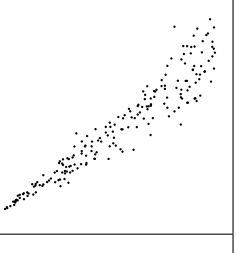
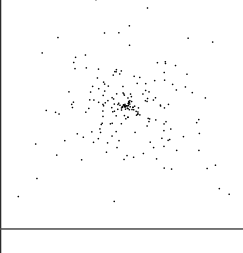
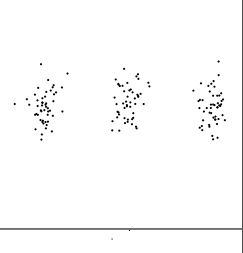
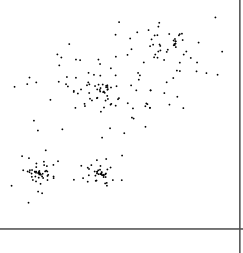
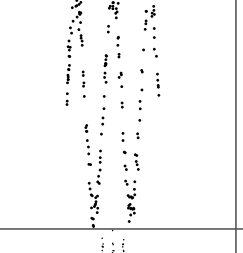
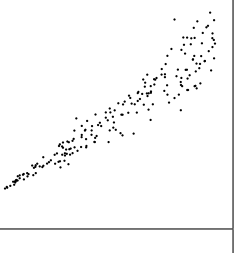
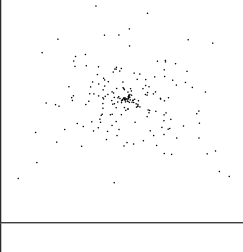
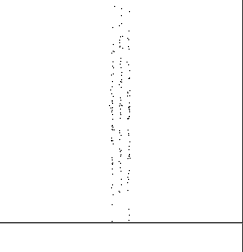
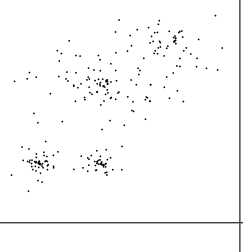
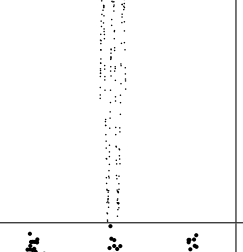
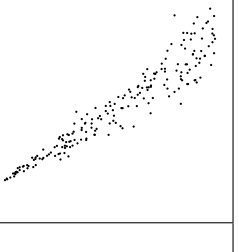
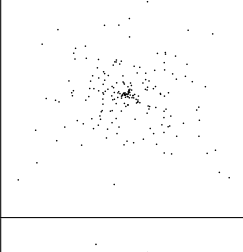
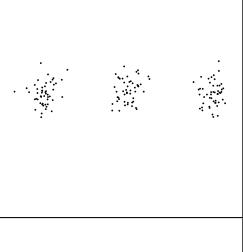
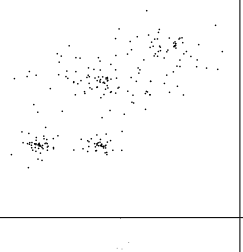
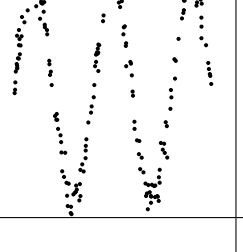
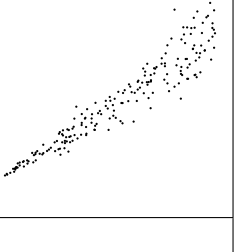
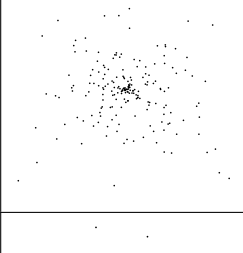
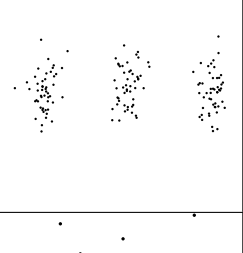
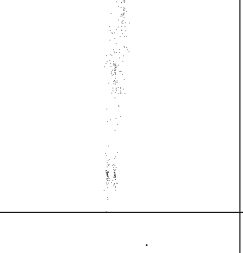
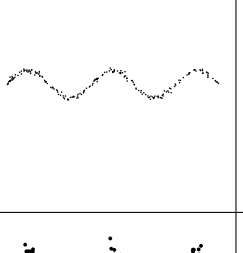
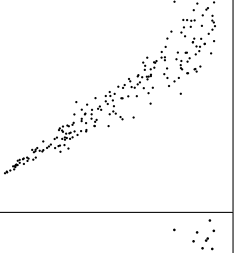
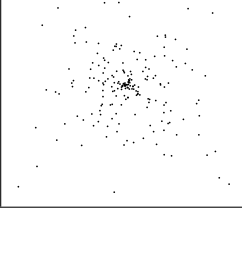
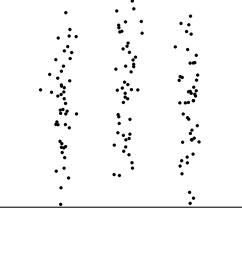
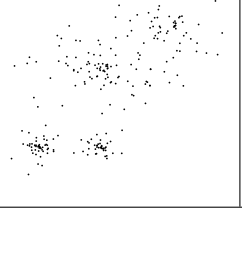
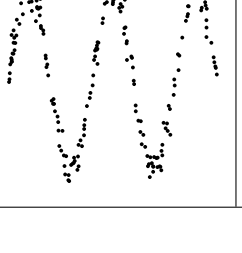
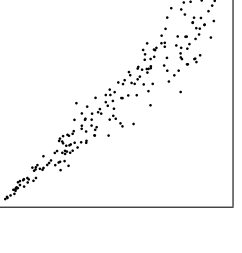
In another test, we generated a collection of four clusters of varying sizes and with different standard deviations, see the middle column of Table 1. In this case all methods produced nice outputs, except for the maximization of the smallest angle, which stretched the scatter plot in an unacceptable way. Due to its nature as a max-min-criterion such a behavior occurs easily; there only needs to be a small substructure in which a very small angle exists at any aspect ratio.

5.3 Results on Point Sets Showing Trends

A second structure that scatter plots help reveal and visualize are *trends* (for example, over time) and functional dependencies. Here we created instances with points randomly scattered around linear, quadratic, and sine functions; the standard deviation from the value of the function was either chosen as a constant or proportional to the function value.

Two such instances are included in Table 1. For the rough linear trend shown in the last column, all methods achieved good results, that is, the slope of the trend was close to 45° . In contrast, the results for a sine-like trend (fourth column) heavily depended on the optimization criterion. The aspect ratios achieved by minimizing the smallest angle, minimizing total edge length, and minimizing uncompactness allow us to recognize the sine function. In contrast, the drawings achieved by minimizing the sum of squared angles, maximizing compactness, and by using the standard deviation criterion are stretched too much in y-direction. This makes it hard to see whether the trend is a sine function or a polyline of a few straight-line segments. The drawing resulting from maximizing the mean inradius is stretched extremely and does not allow the trend to be seen at all.

Table 1. Test results for seven optimization criteria on five generated instances (outputs scaled to fit into the boxes).

	normal distribution	three clusters	four mixed clusters	noisy sine	rough trend
min. total length					
min. uncompact.					
max. compact.					
max. mean inradius					
min. squared angles					
max. min. angle					
standard deviation					

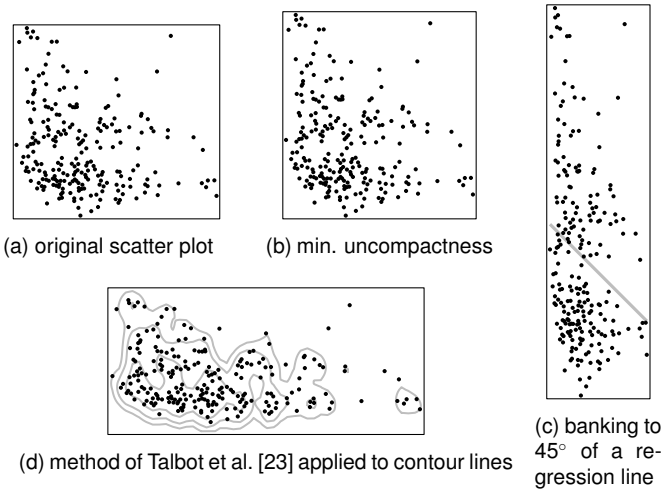


Fig. 7. A scatter plot showing velocities of DNA with different molecule lengths [27] in aspect ratios computed by different methods.

5.4 Results on Real Scatter Plots

We also tested our method on nine scatter plots used in *Nature* articles [4, 13, 23, 27]. Both minimizing total edge length and minimizing total uncompactness resulted in scale factors between 0.64 and 1.27 relative to the aspect ratios chosen by the authors. We consider this quite close to 1. For one of the instances, Fig. 7 shows the original scatter plot, the result of minimizing uncompactness, the result of applying the method of Talbot et al. [23] to contour lines generated with R's `kde2d`, and the result of banking a regression line to 45° as suggested by Cleveland et al. [3]. The results for all instances we processed can be found in the supplementary material. Before applying any of the methods to an instance, we prescaled the instance by a factor of 2. This ensured that we did not overrate the quality of a method that tends to select aspect ratios close to the input aspect ratio.

Our experiment shows that minimizing total edge length and minimizing uncompactness yields scatter plots similar to the scatter plots published in *Nature*. In particular, with these two measures, our method works well on both clustered data *and* data containing trends. An intuitive explanation for this is that, on data containing a trend, the triangulation is dominated by edges following the direction of the trend line, thus decreasing the total length of triangulation edges (which is rewarded with both measures) has an effect similar to banking the trend line to 45° . With clustered data, on the other hand, the triangulation contains for each point the star connecting that point with its natural neighbors. Minimizing the edge lengths thus means that the neighbors are attracted by the central point and thus clusters become nicely compacted rather than unnaturally stretched in one direction. Banking a regression line to 45° yields nice results if the data contains a clear trend (e.g., columns 1 and 2 in Table 1 of the supplementary material) but clearly fails on clustered data (column 2 of Table 2). The result of the method based on contour lines turned out to yield results close to the input aspect ratio, see the last row of Tables 1 and 2.

5.5 Empirical Study

To assess our algorithms, we conducted a study in which we asked every participant to choose the aspect ratios of the same 18 scatter plots. The study was based on a Java applet, which is accessible on-line¹. We received 64 submissions in total, 34 from colleagues, 11 from students, 16 from others, and 3 did not answer to questions regarding themselves. Among the 95% of the participants who did answer these questions, the average age was 28 years and 87% were male.

5.5.1 Experimental Set Up

In brief instructions, we told the participants that, for each of the point sets they will be shown, they are supposed to control the aspect ra-

tio with a slider and to accept the aspect ratio that allows structures, clusters, or trends to be perceived best or, in general, gives the best overview of the data. Additionally, we showed a set of points forming a smiley and suggested that the best visualization of the point set is the one in which the smiley has a circular shape.

After the instructions, we showed 18 point sets in succession. Each point set was initially displayed such that it fitted a square and the slider was in its middle position. Then, by changing the slider position, the participants could select the aspect ratio from the range between 1 : 3 and 3 : 1. We chose the test instances such that this range covered all reasonable values of the aspect ratio. Before participants could confirm their preferred aspect ratio for a given instance, the applet forced them to check all ratios by moving the slider at least once from its rightmost to its leftmost position. Participants could, however, at any time skip the current point set and proceed with the next one.

We designed the first two point sets of the test to accustom the participants with their task. (We showed the instance with the smiley and a point set that, when appropriately scaled, formed a regular pentagon.) Then, we showed four automatically generated instances with clusters where the points were normally distributed around the cluster centers. These instances included an instance with a single cluster, two instances with multiple clusters side by side, and an instance with multiple clusters intersecting each other (middle column of Table 1). The next nine instances were automatically generated by scattering points around graphs of functions, for example, a straight line, a sine curve, and a parabola. Finally, we showed three instances from Fisher's iris data set [6], which is frequently used to assess classification methods.

After the last instance was completed, we asked the participants to rate the difficulty of the test between "very easy" (1.0) and "very difficult" (5.0). On average, the test was found to be "fair" (2.7). Each of the 18 instances was solved by at least 58 of the 64 participants.

5.5.2 Assessment of Results

For each instance, we computed the *geometric* mean $\mu_g(s) = \sqrt[s_1 \cdot \dots \cdot s_k]{s_1 \cdot \dots \cdot s_k}$ of the values s_1, \dots, s_k that the users had selected for the scale factor s (determining the new coordinates of a point (x, y) as $(s \cdot x, y/s)$; see Sect. 3). To clarify why we must use the geometric mean instead of the more common arithmetic mean $\mu(s) = (s_1 + \dots + s_k)/k$, we note that, instead of analyzing values chosen for s , we could just as well analyze values chosen for $1/s$. Therefore, the mean of 2 and $1/2$ (or more generally, of a number a and its reciprocal $1/a$) should be 1 (rather than 1.25). Moreover, if the mean of two numbers a and b is c , then the mean of $1/a$ and $1/b$ should be $1/c$. This is ensured with the geometric mean. Accordingly, to assess the variation in the values selected for s , we computed for each instance the *geometric* standard deviation $\sigma_g(s)$ of s as $\sigma_g(s) = \exp(\sigma(\ln s))$, where $\sigma(\ln s)$ is the common (i.e., arithmetic) standard deviation of the values $\ln s_1, \dots, \ln s_k$. To clarify the meaning of $\sigma_g(s)$, we note that, if the values $\ln s_1, \dots, \ln s_k$ follow a normal distribution, the interval $[\mu_g(s)/\sigma_g(s), \sigma_g(s) \cdot \mu_g(s)]$ contains roughly 68% of all samples.

On average over all instances, the geometric standard deviation was 1.47, which means that there was a rather low agreement among the participants. While the lowest variation ($\sigma_g(s) = 1.18$) was found in the solutions for the set of points that form a smiley, the highest variation ($\sigma_g(s) = 1.81$) occurred for a set of points containing three clusters whose centers lie on a common horizontal line.

We compared the results of the empirical study with the results of our algorithm, which we tested with all six objective functions that we defined in Sect. 3. More precisely, for each of the 18 instances and each scale factor s_{opt} that we obtained by optimizing one of our objective functions, we computed the *geometric standard score*

$$z = \frac{\ln(s_{\text{opt}}/\mu_g(s))}{\ln \sigma_g(s)} = \frac{\ln s_{\text{opt}} - \mu(\ln s)}{\sigma(\ln s)}, \quad (2)$$

where $\mu(\ln s)$ is the arithmetic mean of the values $\ln s_1, \dots, \ln s_k$.

The geometric standard score z tells us by how many values of the standard deviation $\sigma(\ln s)$ the result $\ln s_{\text{opt}}$ of our algorithm is above or below the mean $\mu(\ln s)$. Therefore, its absolute value $|z|$ is a reasonable quality measure. The average \bar{z} of $|z|$ over all instances then

¹www1.informatik.uni-wuerzburg.de/scatterplots

yields a single quality measure for each objective function that we optimized with our algorithm. Additionally, we used the same approach to measure the quality of the simple standard-deviation-based method that we introduced in Equation (1). Table 2 summarizes our results.

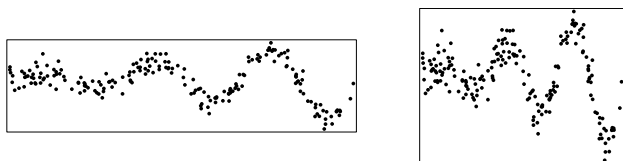
Table 2. Absolute values of geometric standard scores computed based on geometric means and geometric standard deviations that resulted from the empirical study and scale factors that resulted from our algorithm. Minima, maxima, and averages were computed based on 18 test instances. The last row shows the results of the simple standard-deviation-based method defined in Equation (1).

method	minimum	maximum	average (\bar{z})
max. min. angle	0.02	3.47	0.82
max. mean inradius	0.02	6.62	1.74
min. total length	0.05	1.75	0.48
min. squared angles	0.06	3.08	0.62
max. comp.	0.01	4.40	0.81
min. uncompact.	0.05	1.59	0.45
standard deviation	0.06	2.64	0.84

From Table 2 we conclude that, on average, minimizing the total edge length and minimizing the mean uncompactness of the triangles yield the best results. With both objective functions, \bar{z} is less than 0.5. To understand the meaning of this value, we also computed \bar{z} based on the results of each individual participant, that is, for the aspect ratios selected by each participant we assessed the distances to the mean aspect ratios. For 38 of the 64 participants (i.e., for 59%) we obtained $\bar{z} \geq 0.5$, meaning that our algorithm performed better than most of the participants (assuming that the aim was to get close to the mean aspect ratios over all participants). Maximizing the mean inradius seems inappropriate, since $\bar{z} = 1.74$ is very high. With all other objective functions, however, our method produces somehow reasonable results.

Using the simple standard-deviation-based method resulted in $\bar{z} = 0.84$, which shows that this method may be useful. However, when optimizing any of our objective functions (except when maximizing the mean inradius) our method resulted in a smaller value for \bar{z} .

With our best method (i.e., minimizing the mean uncompactness of the triangles) the highest value $|z|$ attained for one instance was 1.59. That instance consists of a set of points that are scattered around the graph of the function $f: x \mapsto x \sin x$, see Fig. 8. When comparing the scatter plot favored by the users (Fig. 8a) and the scatter plot yielded by our algorithm (Fig. 8b) we observe that, though the aspect ratios of both scatter plots are quite different, they both allow the functional relationship between the two variables to be perceived.



(a) scaled to the mean aspect ratio chosen by the participants of our user study (b) scaled to the best aspect ratio according to the uncompactness criterion

Fig. 8. An instance used in our study.

We also applied the method of Talbot et al. [23], which for a given line chart yields the aspect ratio that minimizes the length of the displayed line, to the nine instances in which we scattered the points around the graphs of functions. Since the method of Talbot works well for line charts, we exploited our knowledge of the functions that we used to generate the point sets. More precisely, we applied the method of Talbot et al. to piecewise-linear approximations of the graphs of the nine functions. Again, we compared the results with the aspect ratios selected by the participants of our study. On average over the nine instances, the method of Talbot et al. (with additional knowledge)

achieved $\bar{z} = 0.41$, whereas we obtained $\bar{z} = 0.55$ when we applied our point-based method with the mean uncompactness criterion to the nine point sets (without additional knowledge). This means that, if we already know the functional relationship between the two variables of the scatter plot, we should use this knowledge and apply the method of Talbot et al. to the graph of the function. If we do not know a functional relationship, however, our point-based method still achieves aspect ratios that are almost as good as those selected with the use of the prior knowledge. We repeated this experiment with the arc-based method of Cleveland [2]. On our instances, this yielded almost the same results as the method of Talbot et al.

5.6 Runtime

As, in praxis, a user who wants to draw a scatter plot is not willing to wait long for the result, the runtime of our method is important. We measured runtimes using our implementation on a standard PC with 4 GB RAM and an Intel Core2 Duo CPU with 3 GHz. We measured the time needed for finding an aspect ratio that is a 1.05-approximation with respect to minimizing the total edge length. To this end, we generated noisy sine functions as well as clustered instances similar to the one shown in the middle column of Table 1 consisting of a growing number of points; see Table 3.

Table 3. Runtimes (in ms) for minimizing the total edge length on instances of different sizes.

number of points	50	100	200	400	600	800	1 000
noisy sine	45	59	167	528	1 113	2 139	3 574
four mixed clusters	51	50	259	795	1 947	3 516	5 659

In the worst case, our method may have to traverse $\Theta(n^2)$ Delaunay triangulations. On the worst-case instance $P_{wc}(n)$ (see Sect. 4.1), this took about 10 seconds for $n = 1000$. On typical instances, however, our method is much faster. For example, for the clustered instance shown in the middle column of Table 1, which contains 240 points, an optimum solution (maximizing the smallest angle) was found in one second; a 1.05-approximate solution for the better criterion of minimizing the total edge length was computed in 244 ms. (For instances with more than 1000 points, legibility and runtime become issues. To improve the runtime, we simply suggest applying our algorithm to randomly selected subsets of the points.)

As only the evaluation of the quality measure differs, obtaining approximate solutions for the remaining criteria took nearly the same time. Optimizing the aspect ratio of the other instances with a similar number of points was even faster. We conclude that our method is fast enough for practical applications.

6 CONCLUSION

We have presented a new method that automatically chooses the aspect ratio of a scatter plot. The basic idea behind our method is to select the aspect ratio such that the resulting scatter plot has a nice Delaunay triangulation. We defined “nice” by six different measures, which are motivated by properties of the Delaunay triangulation, by related work on the problem of choosing the aspect ratio of a line chart, or by previous research on shape analysis.

We conclude that, in terms of quality, our method performs particularly well when minimizing the *total edge length* of the Delaunay triangulation or the *mean uncompactness* of the Delaunay triangles. For both measures, our algorithm yields scatter plots in which clusters or trends are clearly visible. The suitability of these two measures was confirmed by applying them on scatter plots published in *Nature* and by comparing them to aspect ratios chosen by subjects in a user study.

An idea for future research is to assess the quality of a scatter plot by asking users to solve certain tasks. An interesting algorithmic problem that our research brings up is to compute aspect ratios that optimize favorable properties of other neighborhood graphs than the Delaunay triangulation, for example, the Gabriel graph.

REFERENCES

- [1] G. Albers, L. J. Guibas, J. S. Mitchell, and T. Roos. Voronoi diagrams of moving points. *Int. J. Comput. Geom. Appl.*, 8(3):365–380, 1998.
- [2] W. S. Cleveland. A model for studying display methods of statistical graphics. *J. Comput. Graph. Statist.*, 2(4):323–343, 1993.
- [3] W. S. Cleveland, M. E. McGill, and R. McGill. The shape parameter of a two-variable graph. *J. Am. Stat. Assoc.*, 83(289–300), 1988.
- [4] D. Dawson and K. Reid. Fatigue, alcohol and performance impairment. *Nature*, 388(6639):235–235, 1997.
- [5] M. J. Dry, D. J. Navarro, K. Preiss, and M. D. Lee. The perceptual organization of point constellations. In *Proc. 31st Annual Conference of the Cognitive Science Society*, pages 1151–1153, 2009.
- [6] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.
- [7] S. Fortune. Voronoi diagrams and Delaunay triangulations. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 513–528. CRC Press, Boca Raton, FL, U.S.A., 2004.
- [8] M. Friendly and D. Denis. The early origins and development of the scatterplot. *Journal of the History of the Behavioral Sciences*, 41(2):103–130, 2005.
- [9] K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18(3):259–278, 1969.
- [10] L. J. Guibas and J. S. Mitchell. Voronoi diagrams of moving points in the plane. In *Proc. 17th Int. Workshop Graph-Theor. Concepts Comput. Sci. (WG'91)*, pages 113–125, 1991.
- [11] J. Heer and M. Agrawala. Multi-scale banking to 45°. *IEEE T. Vis. Comput. Gr.*, pages 701–708, 2006.
- [12] J. F. W. Herschel. On the investigation of the orbits of revolving double stars. *Memoirs of the Royal Astronomical Society*, pages 171–222, 1833.
- [13] C. Johnson. Species extinction and the relationship between distribution and abundance. *Nature*, 394(6690):272–274, 1998.
- [14] T. Lambert. The Delaunay triangulation maximizes the mean inradius. In *Proc. 6th Canadian Conf. Comput. Geom. (CCCG'94)*, pages 201–206, 1994.
- [15] A. M. MacEachren. Compactness of geographic shape: Comparison and evaluation of measures. *Geografiska Annaler. Ser. B, Human Geogr.*, 67(1):53–67, 1985.
- [16] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. <http://www.r-project.org/>.
- [17] T. Roos. Voronoi diagrams over dynamic scenes. *Discrete Appl. Math.*, 43(3):243–259, 1993.
- [18] N. Rubin. On topological changes in the Delaunay triangulation of moving points. In *Proc. 28th ACM Symp. Comput. Geom. (SoCG'12)*, pages 1–10, 2012.
- [19] D. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley Series in Probability and Statistics. Wiley, New York, USA, 1992.
- [20] M. Sharir and P. K. Agarwal. *Davenport-Schinzel sequences and their geometric applications*. Cambridge University Press, 1995.
- [21] R. Sibson. Locally equiangular triangulations. *The Computer Journal*, 21(3):243–245, 1978.
- [22] K. A. Stevens. Computation of locally parallel structure. *Biological Cybernetics*, 29(1):19–28, 1978.
- [23] J. Talbot, J. Gerth, and P. Hanrahan. Arc length-based aspect ratio selection. *IEEE T. Vis. Comput. Gr.*, 17(12):2276–2282, 2011.
- [24] J. Talbot, J. Gerth, and P. Hanrahan. An empirical model of slope ratio comparisons. *IEEE T. Vis. Comput. Gr.*, 18(12):2613–2620, 2012.
- [25] G. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12(4):261–268, 1980.
- [26] E. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.
- [27] W. Volkmuth and R. Austin. DNA electrophoresis in microlithographic arrays. *Nature*, 358(6387):600–602, 1992.
- [28] J. Wagemans, J. H. Elder, M. Kubovy, S. E. Palmer, M. A. Peterson, M. Singh, and R. von der Heydt. A century of Gestalt psychology in visual perception: I. Perceptual grouping and figure-ground organization. *Psychological Bulletin*, 138(6):1172–1217, 2012.