Algorithmic Aspects of Proportional Symbol Maps

Sergio Cabello

Herman Haverkort

Marc van Kreveld

Bettina Speckmann

Department of Information and Computing Sciences, Utrecht University

Technical Report UU-CS-2008-008

www.cs.uu.nl

ISSN: 0924-3275

Algorithmic Aspects of Proportional Symbol Maps *

Sergio Cabello[1,†] Herman Haverkort[2] Marc van Kreveld[3] Bettina Speckmann[2]

[1] Department of Mathematics, Institute for Mathematics, Physics and Mechanics, Ljubljana, Slovenia.
sergio.cabello@fmf.uni-lj.si
[2] Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands.
cs.herman@haverkort.net and speckman@win.tue.nl
[3] Institute for Information and Computing Sciences, Utrecht University, The Netherlands.
marc@cs.uu.nl

Abstract

Proportional symbol maps visualize numerical data associated with point locations by placing a scaled symbol—typically an opaque disk or square—at the corresponding point on a map. The area of each symbol is proportional to the numerical value associated with its location. Every visually meaningful proportional symbol map will contain at least some overlapping symbols. These need to be drawn in such a way that the user can still judge their relative sizes accurately.

We identify two types of suitable drawings: *physically realizable drawings* and *stacking drawings*. For these we study the following two problems: Max-Min—maximize the minimum visible boundary length of each symbol—and Max-Total—maximize the total visible boundary length over all symbols. We show that both problems are NP-hard for physically realizable drawings. Max-Min can be solved in $O(n^2 \log n)$ time for stacking drawings, which can be improved to $O(n \log n)$ time when the input has certain properties. We also implemented several methods to compute stacking drawings: our solution to the Max-Min problem performs best on the data sets considered.

1 Introduction

Proportional symbols maps, which are also known as *graduated symbol maps*, are a well established cartographic tool to visualize quantitative data that is associated with specific (point) locations. According to Clarke [3] they are even the only method to visualize numerical data associated with points.

A symbol, most commonly a disk or a square, is scaled such that its area corresponds to the data value associated with a point and then placed at exactly that point on a geographic map. The spatial distribution of the data can then be observed by studying the spatial distribution of the differently sized symbols. Typical data that are visualized in this way include the magnitude of earthquakes (see Fig. 1), the production of oil wells, or the

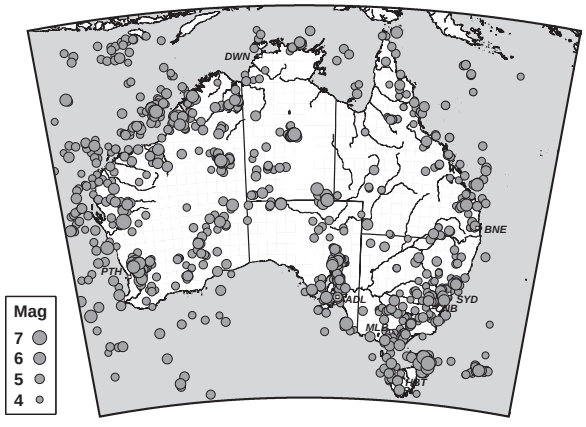


Figure 1: A classed proportional symbol map depicting Australian earthquakes of size ≥ 4.0 on the Richter scale [18].

*A preliminary version of this paper appeared in the *Proceedings of the 14th European Symposium on Algorithms*, pp. 720-731, LNCS 4168, Springer Verlag, 2006.

†Research partially supported by the Slovenian Research Agency, project J1-7218.

temperature at weather stations. One distinguishes between *true point data* and *conceptual point data*. The former is actually measured at a point whereas the latter is aggregated over an area but is conceived as being located at a point. Data collected within a city are commonly considered to be true point data since a city is represented as a point on most maps. Conceptual point data include data that are collected in a province or state and that are displayed at a location that is representative of the region in question. In this paper we consider only proportional symbol maps for true point data.

A proportional symbol map communicates its message via the sizes of its symbols—both the actual size of the symbols and the ratio between symbol sizes. A large body of theoretical work and user studies discuss which sizing communicates the difference between quantities in the most effective way. See the books by Dent [6] and Slocum et al. [20] for an extensive overview. Here we mention only three basic types of symbol scaling. The so-called *mathematical scaling* sizes the areas of the symbols in direct relation to the data. However, humans find it difficult to judge the relative sizes of area symbols accurately, specifically, the sizes of larger symbols are often underestimated. *Perceptual scaling* tries to compensate for this by enlarging larger symbols beyond their mathematically correct size. Finally, *range grading* subdivides the data into classes and produces a map where all data in one class are represented by a single symbol size. The resulting map is often referred to as a *classed map* (see for example Fig. 1).

While it is commonly agreed upon that a map should appear "neither 'too full' nor 'too empty'" [20] it is unclear how much the symbols on a proportional symbol map should overlap. Small symbols create little or no overlap but spatial patterns are difficult to detect. On the other hand, large symbols result in a cluttered map where it is difficult to identify and judge individual symbols. Determining the ideal size for the symbols is a major issue when constructing proportional symbol maps, but every "good" map will contain at least some overlapping symbols (see the discussion in Slocum et al. [20]).

In principle any two-dimensional shape can be used as a symbol on a proportional symbol map. However, circles (transparent) and disks (opaque) are used most frequently, since they are visually stable, they conserve map space, and users prefer them. Also squares and triangles are occasionally seen. More complex, pictographic symbols, for example beer mugs or oil wells, or three-dimensional symbols such as spheres, make for a catchy and memorable map, but users have significant problems to judge their relative sizes accurately. In this paper we concentrate on geometric symbol shapes, in particular disks, squares, triangles and other convex shapes.

If the symbols are transparent, then the user can see through overlapping symbols and still judge their sizes more or less accurately. Opaque symbols, on the other hand, obscure anything that lies below them. Nevertheless, studies [10] have shown that users (slightly) prefer opaque symbols since they contrast better with the underlying geographic map. Not too surprisingly other studies [11] show that there is a strong correlation between the amount of overlap of opaque symbols and the error that occurs when judging their respective sizes.

Clearly there are many different ways to arrange opaque symbols with respect to each other and any choice of (partial) order makes some symbols more visible than others. In this paper we address the algorithmic question how to arrange a given set of overlapping disks, squares or other convex symbols such that all of them can be seen as well as possible.

Definitions and notation. Before we can formally state the problem we first need to introduce some definitions and notation. To simplify the presentation we give all definitions for disks, but they naturally extend to opaque squares or other shapes. Let S be a set of n disks $D_1, \ldots, D_n$ in the plane. We denote by $\mathcal{A}$ the arrangement formed by the boundaries of the disks in S. A *drawing* $\mathcal{D}$ of S is a subset of the arcs and vertices of $\mathcal{A}$ which is drawn on top of the filled interiors of the disks in S. A drawing is *bounded* if it includes the boundary of the union of the disks in S.

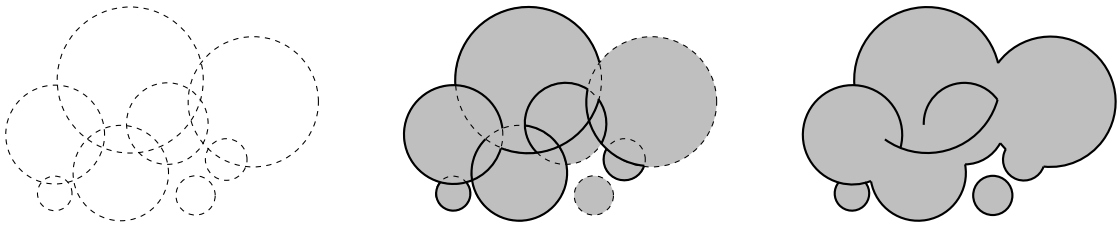


Figure 2: An arrangement $\mathcal{A}$, a drawing with $\mathcal{A}$ visible, a bounded drawing.

Not every drawing is suitable for the use on a proportional symbol map. A suitable drawing needs to be bounded. It should be locally correct at the vertices: a drawing is locally correct at a vertex v, formed by the intersection of the boundaries of two disks D_i and D_j, if locally around v the drawing corresponds to stacking D_i onto D_j or vice versa. Furthermore, a suitable drawing must have only correct faces: a face of the drawing is correct if there is an order in which all disks in S that contain the face can be drawn on top of each other such that the face appears. We call drawings that satisfy these conditions *face correct*.

Figure 3: A bounded, vertex correct drawing, which is not face correct, with and without $\mathcal{A}$ visible (left), a bounded, face correct drawing with and without $\mathcal{A}$ visible (right).

Figure 3 shows that even a face correct drawing can still have an "Escher-like" quality which we would like to avoid on a proportional symbol map. Hence we need to enforce even stronger requirements on what constitutes a proper drawing. We consider two types of drawings.

Physically realizable drawings. A face correct drawing is physically realizable if and only if for every face f of the arrangement $\mathcal{A}$ there exists a total order on the disks in S_f (the disks in S that contain f) such that the topmost disk is visible and the orders associated with any two faces of $\mathcal{A}$ do not conflict. That is, the order in which the disks in S are stacked upon each other is uniquely determined at every face of $\mathcal{A}$ and no two of such orders conflict. In particular, any two or more disks that have a common intersection have a unique ordering. The orders of the disks for all faces of $\mathcal{A}$ immediately imply which arcs of $\mathcal{A}$ are part of the drawing.

We observe that this definition is in fact equivalent to the following. We associate a *pr-disk* D_i' with every disk D_i in S. D_i' is a surface patch that is the image of a continuous function of the points in the input disk, that is, $(x, y) \in D_i$ maps to $(x, y, f_i(x, y))$ where $f_i(\cdot, \cdot)$ is continuous. The boundary of D_i' is a closed curve that lies in a cylinder erected vertically on the boundary of D_i. A drawing $\mathcal{D}$ is physically realizable if functions $f_1, \ldots, f_n$ exist so that the pr-disks $D_1', \ldots, D_n'$ are disjoint and the view vertically down from infinity is $\mathcal{D}$. That is, if we imagine that we are working with actual physical disks then we are allowed to warp them in a "Dali-like" fashion, but we cannot cut them.

Stacking drawings. A stacking drawing is a natural restriction of a physically realizable drawing and also the one most frequently found on proportional symbol maps. A physically realizable drawing $\mathcal{D}$ is a stacking drawing if there exists a total order on the disks in S such that $\mathcal{D}$ is the result of stacking the disks in this order. We call such a total order a *stacking order*.

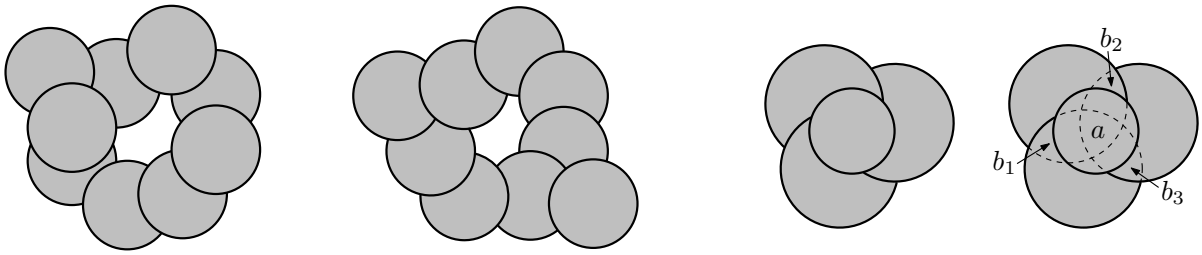


Figure 4: A stacking drawing (left), a physically realizable drawing that is not a stacking drawing (middle), a drawing that may seem physically realizable, but is not—any order for face a will conflict with one of b_1, b_2, or b_3 (right).

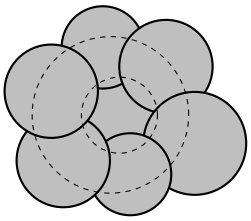


Figure 5: Visible perimeter is more important than visible area.

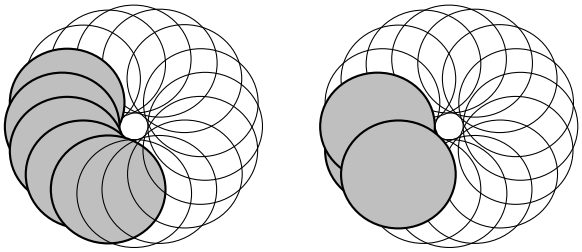


Figure 6: An optimal physically realizable drawing (left), an optimal stacking drawing for the same disks (right).

Quality of a drawing. Intuitively, a good drawing should enable the viewer to see at least some part of all symbols and to judge their sizes as correctly as possible. The accuracy with which the size of a disk can be judged is proportional to the portion of its boundary that is visible. This leads us to the following two optimization problems. Assume that we are given a set S of n opaque symbols $S_1, \ldots, S_n$.

Max-Min: Find a physically realizable or a stacking drawing that maximizes the minimum visible boundary length of each symbol, that is, $\max \min_{1 \le i \le n} \{\text{visible length of the boundary of } S_i\}$.

Max-Total: Find a physically realizable or a stacking drawing that maximizes the total visible boundary length over all symbols.

Figure 5 illustrates why we consider only visible boundary length and not visible area of symbols. The boundary of the center disk is completely covered but a significant part of its area is still visible. It is, however, impossible to judge its size or to determine the location of its center. Figure 6 shows that a stacking drawing can be arbitrarily much worse than a physically realizable drawing with respect to the Max-Min problem. At least half of the boundary of every disk in Figure 6 (left) is visible, whereas the lowest disk in any stacking drawing is covered by its two neighbors and hence has only a very short visible boundary.

Formal problem statement. Assume that we are given a set S of n opaque homothetic disks, squares or other convex symbols that may overlap. Construct a physically realizable drawing or a stacking drawing for the elements of S that either maximizes the minimum visible boundary of each symbol (Max-Min) or maximizes the total visible boundary of all symbols (Max-Total).

Results. We show in Section 2 that for physically realizable drawings both the Max-Min and the Max-Total problems are NP-hard. For stacking drawings the Max-Min problem can be

solved in $O(n^2 \log n)$ time. If no point in the plane is covered by more than $O(1)$ symbols, then it can be solved in $O(n \log n)$ time. If the symbols are unit-size squares it can be solved in $O(n \log n)$ time, regardless of how many squares overlap in a point. These algorithmic results are presented in Section 3. The status of the Max-Total problem for stacking drawings is open. We performed experiments to compare the results of four different methods that compute a stacking drawing. One of these is our solution to the Max-Min problem, and this one performs best on the data sets considered. These results are presented with various tables and figures in Section 4.

2 NP-hardness

We show that the Max-Min and the Max-Total problems are NP-hard for physically realizable drawings of disks. For the NP-hardness proofs, we have to restrict ourselves to the RAM model of computation, and therefore we have to be careful that our reductions only use disks with integer radii and centers located at integer coordinates. Note that the visible perimeter of a drawing is a sum of the lengths of some circular arcs, and therefore, it is unclear if the problems we are considering belong to the class NP. This is not surprising, since many basic geometric problems are not known to be in NP [5, 7]. Both the reduction for Max-Min and the reduction for Max-Total are from planar 3-SAT, which was proved NP-hard by Lichtenstein [15]. Since the ideas are standard and often used (see for example [1, 2, 8, 14]), our discussion concentrates on the gadgets.

2.1 Max-Min is NP-hard

For any given instance $\mathcal{I}$ of planar 3-SAT we define a set S of disks of perimeter 1 with the following property: The disks have a physical realization with a free perimeter of at least 3/4 per disk if and only if the planar 3-SAT formula is satisfiable. The construction we give is geometrically intuitive, but uses non-integer values for coordinates and radius. We explain later how an equivalent construction can be made in polynomial time and with integer values, hence providing a valid NP-hardness reduction.

The set S consists of a set of disks, a *gadget*, for every variable, every clause, and every literal in a clause of $\mathcal{I}$. Gadgets for literals are called *channels* and connect variable and clause gadgets.

Variable gadgets. We say that two disks overlap for a fraction f if a fraction f of the boundary of one disk is covered by the other disk. Since we use disks of perimeter one, a fraction f of its boundary has length f.

A Boolean variable x_i is represented by an even cycle $S(x_i)$ of disks, as shown in Figure 7. Any two adjacent disks overlap for 1/8 or 1/4, such that any disk overlaps for 1/8 with one neighbor and for 1/4 with the other neighbor. Hence, to achieve that each disk has 3/4 of its perimeter visible, the cycle must be either clockwise overlapping (signifying that x_i is TRUE) or counterclockwise overlapping (signifying that x_i is FALSE).

In the TRUE state, every second disk has 1/4 of its boundary covered by the previous disk in the cycle—if 3/4 of the boundary of each of these disks is to remain visible, no more disks (other than those in the cycle) must cover them. For the other half of the disks in the cycle, only 1/8 of their boundaries are covered and disks outside the cycle may cover another fraction 1/8 of them. In the FALSE state, it is precisely the other set of disks that can be covered for another 1/8.

Clause gadgets. A clause is represented by a single disk.

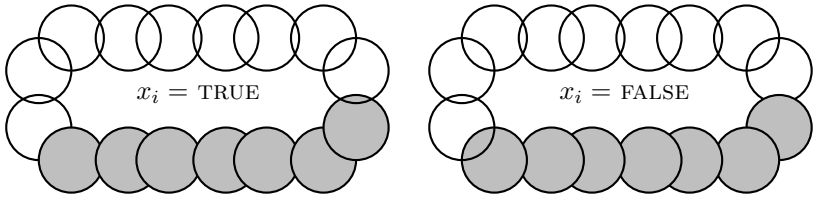


Figure 7: Representation of a Boolean variable and its TRUE and FALSE states.

Channels. Channels represent literals used in clauses. Each channel is a series of disks such that each disk overlaps the next disk for 1/4.

The first disk of the channel overlaps for 1/8 with a disk in a cycle representing a variable: if the channel represents the literal x_i, it overlaps for 1/8 with a disk of $S(x_i)$ that can take another 1/8 overlap in the TRUE state of $S(x_i)$; if the channel represents $\overline{x_i}$, it overlaps for 1/8 with a disk of $S(x_i)$ that can take another 1/8 overlap in the FALSE state of $S(x_i)$. If the variable has enough disks in its cycle, then any number of channels can be connected and in any order for x_i and $\overline{x_i}$.

A channel represents TRUE if the disks in the channel are stacked from bottom to top when traversing the channel from the variable to the clause, leaving only 3/4 of the boundary of the disk at the variable end visible. Hence a channel for x_i can represent TRUE only if the state of $S(x_i)$ is TRUE, and a channel for $\overline{x_i}$ can represent TRUE only if the state of $S(x_i)$ is FALSE.

At a clause like $(\overline{x_i} \vee x_j \vee x_k)$, the channels for $\overline{x_i}$, x_j, and x_k come close and the last disks of the channel each overlap for 1/8 with the disk that represents the clause, see Figure 8. For the clause disk to be uncovered for at least 3/4, at least one of the three channels must represent TRUE so that the last disk of that channel can go under the clause disk.

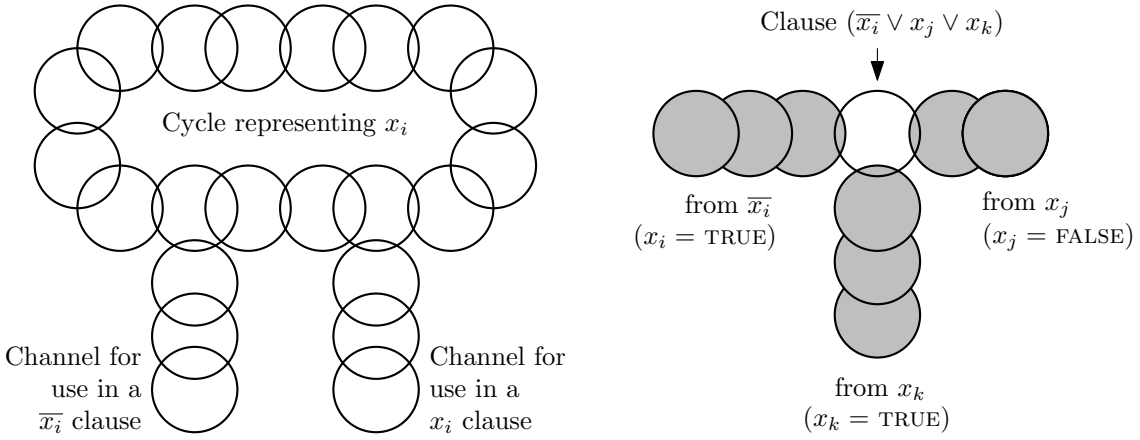


Figure 8: Representation of a clause.

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.*

Proof. Consider an instance $\mathcal{I}$ of 3-SAT and a corresponding set of disks S as described above. We show that S has a physically realizable drawing where at least 3/4 of every disk's boundary is visible if and only if $\mathcal{I}$ is satisfiable.

If $\mathcal{I}$ is satisfiable, fix a Boolean assignment that makes $\mathcal{I}$ true. For each TRUE variable x_i in the assignment, we set the variable gadget $S(x_i)$ and the channels for x_i in their TRUE states, and we draw the channels for $\overline{x_i}$ in the opposite way, that is, with the disks stacked from bottom

to top when traversing the channel from clause to variable, with the last disk at the variable end tucked under $S(x_i)$. For each FALSE variable x_i, we set $S(x_i)$ in its FALSE state, the channels for $\overline{x_i}$ in their TRUE state, and we draw the channels for x_i in the opposite way. We draw each clause disk on top of the TRUE channels arriving at the disk, and under any other channels arriving at the disk. Each clause in $\mathcal{I}$ has at least one literal that is true in the given Boolean assignment, hence at least one of the channels arriving at the clause disk is in its TRUE state and does not cover the disk. One can easily verify that each disk in the complete drawing has 3/4 of its boundary visible.

If S has a physically realizable drawing where at least 3/4 of every disk's boundary is visible, then each variable gadget must be drawn in its TRUE or FALSE state, thus defining a Boolean assignment. Furthermore, each clause gadget must cover the end of at least one channel: for each disk in this channel to be visible for at least 3/4, the channel must be drawn in its TRUE state, and hence, the literal represented by the channel must be made true by the Boolean assignment. It follows that any drawing with at least 3/4 of every disk's boundary visible corresponds directly to a Boolean assignment of variables that fulfills all clauses of $\mathcal{I}$.

Note that only a polynomial number of disks are used in the reduction. Moreover, if it is not possible to achieve a free perimeter of at least 3/4 in all disks, then some disk has a free perimeter of exactly 5/8.

For a reduction that uses integer values, we scale the previous construction by 200π (then all disks have radius 100) and displace each disk so that its center is at the closest point of the integer grid. This construction can be carried out in polynomial time in the RAM model, since for each coordinate of the disks we only need to compute a polynomial number of the bits involved in the previous construction. Displacing the disks changes the length of the arc in the overlap of two disks by at most 4. It is easy to see that the original construction can be done such that the distance between any two disks that do not overlap is at least $\sqrt{2}/200\pi$, so that they are still disjoint after scaling and displacement. It follows that in the new construction each disk overlaps at most three other disks, and therefore the displacement changes the length of the visible boundary of a disk by at most $3 \cdot 4 = 12$. If originally, in a physical realization a disk had visible perimeter 3/4, the visible length of its perimeter after scaling and displacement is at least $200\pi \cdot (3/4) - 12 > 459$. If a disk had a visible perimeter of length 5/8, its visible perimeter is now at most $200\pi \cdot (5/8) + 12 < 405$. Therefore, this new (scaled and rounded) set of disks has a physical realization with free perimeter at least 459 per disk if and only if the 3-SAT formula is satisfiable. □

Note that in the proof, no point in the plane is contained in more than two disks. Furthermore, the proof shows that the decision problem of finding a physically realizable drawing with free perimeter at least $405 = 15 \cdot 27$ is as difficult as the decision problem asking for perimeter at least $459 = 17 \cdot 27$. This shows that we cannot expect to find a (17/15)-approximation to the optimal solution in polynomial time. The factor 17/15 is by no means best possible, and it can be raised with some changes in the construction.

2.2 Max-Total is NP-hard

For the Max-Total problem, we follow the same approach as in the previous proof. We first give a geometric construction, and then discuss how to adapt it to integer values. In the reduction, we use disks of two types: perimeter 1 (unit size) and perimeter 3.

We denote by 1-*bone* (3-*bone*) a pair of unit disks (disks of perimeter 3) that almost coincide. In each bone, there are two choices regarding which one of the two disks goes on top, and this changes on which side of the bone a double boundary becomes visible. Bones are the main instrument in the construction.

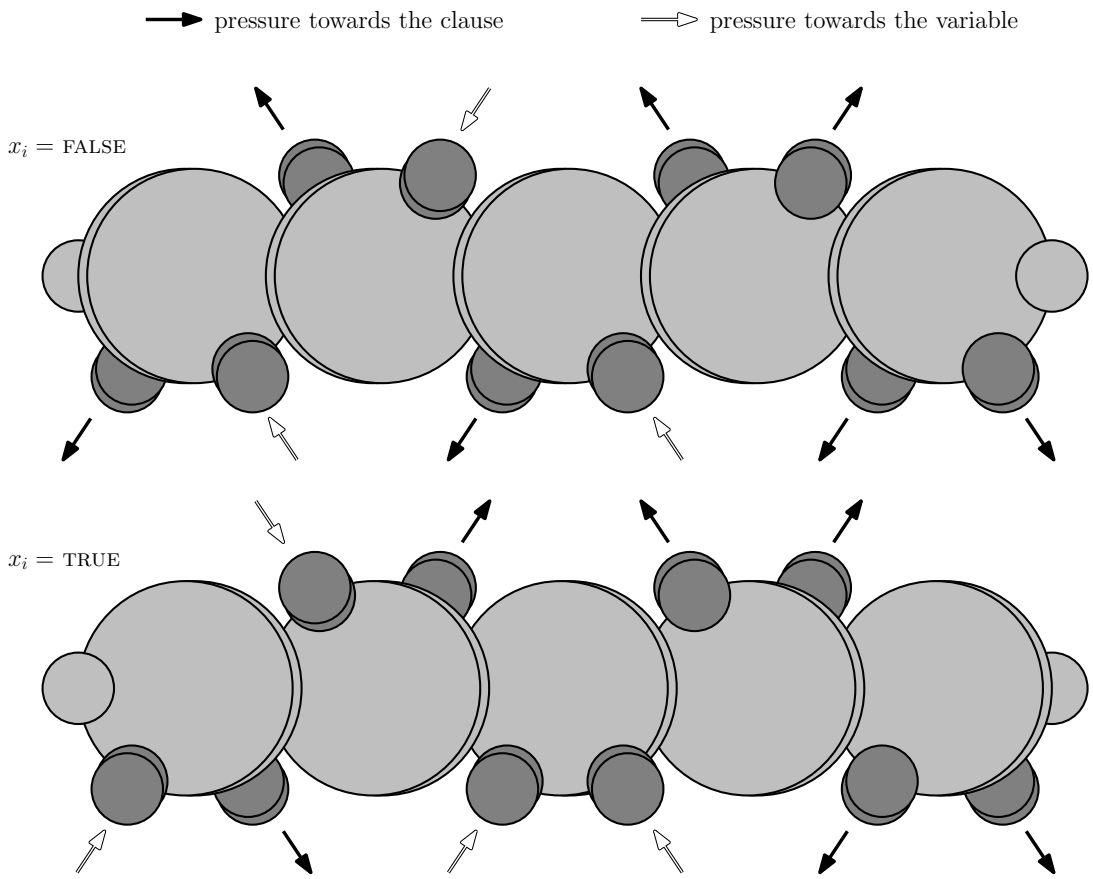


Figure 9: Representation of a Boolean variable. Each spine bone has two readers: one on the left (the *positive* reader) and one on the right (the *negative* reader).

Variable gadgets. We first describe the gadget to represent a Boolean variable; see Figure 9. The gadget consists of a *spine* made of a chain of 3-bones ending with an extra unit disk at each extreme, and a collection of (*variable-*)*readers*. The disks that make up the spine have their centers on a single horizontal line. The readers come in pairs: each 3-bone of the spine has a reader on the left and a reader on the right side, and each reader partially overlaps this 3-bone (and no others). In the drawing, we say that a reader has *pressure towards the variable* when the side of the reader where a double boundary is visible mostly overlaps with the spine; otherwise it has *pressure towards the clause*. Note that we can choose how many spine-bones the variable has, so that we can have many readers.

We next discuss the possible drawings of the variable gadget that maximize the visible boundary. For each individual spine-bone, it does not matter if it is drawn with the double boundary on the left or with the double boundary on the right, since the readers overlapping it make the decision symmetric. However, the two unit disks at the ends of the spine assure that in an optimal drawing, all 3-bones of the spine must have their double boundaries on the same side, as shown in Figure 9. When the double boundaries are on the left we say that the variable gadget is in the FALSE state, and when the double boundaries are on the right it is in the TRUE state.

Assume that a choice of TRUE or FALSE state is made in the variable gadget. To maximize the visible boundary, a reader at a double boundary of a spine bone must be put under the spine bone and have pressure towards the clause. A reader at a single boundary of a spine bone must be put on top of the spine bone and can have pressure in either direction. In an optimal drawing of the variable in a TRUE state, the left-side readers can have pressure in any direction

while the right-side readers must have pressure towards the clause. In an optimal drawing of the variable in a FALSE state, the left-side readers must have pressure towards the clause while the right-side readers can have pressure in any direction. This implies that if any left-side reader has pressure towards the variable the variable must be true, and if any right-side reader has pressure towards the variable the variable must be false. Therefore we call the left-side readers *positive* readers and the right-side readers *negative* readers.

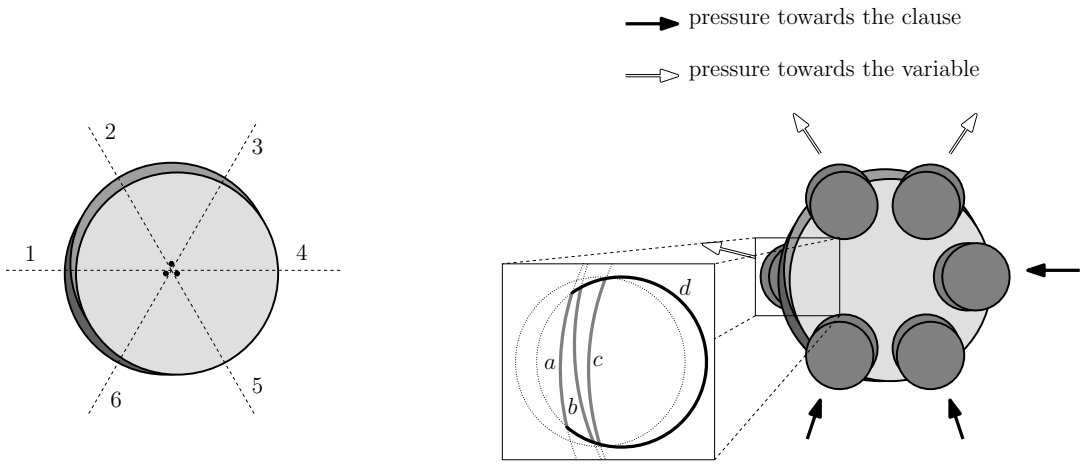


Figure 10: The three large disks in a clause gadget (left), the dots mark their centers. A possible optimal drawing of the clause (right).

Clause gadgets. We now describe the gadget that represents a clause. It consists of three disks of perimeter 3 symmetrically placed and with their centers very close together; see Figure 10 (left). Along each of the 6 dotted directions marked in the figure, we place a 1-bone, called (*clause-*)*reader*, with both of its disks centered on a line through the center of the clause; see Figure 10 (right). The 1-bones are placed carefully, as follows. For a 1-bone, consider its inner disk D (the disk closest to the center of the clause). From the three large disks, let a, b and c be the parts of their boundaries that overlap with D. Let d be the part of the boundary of D that overlaps with the union of the three large disks; see the zoomed part of Figure 10 (right). Let $|a|, |b|, |c|$ and $|d|$ be the lengths of these arcs. We place the 1-bone such that $\max(|a|+|b|, |a|+|c|, |b|+|c|) < |d| < |a|+|b|+|c|$. (As depicted in Figure 10, this can be achieved by placing the centers of the three large disks very close together, because then the arcs a, b and c have almost identical lengths.) For a clause-reader, we say that it has *pressure towards the variable* if most of its double boundary does *not* overlap with the three center disks of the clause; otherwise, we say that it has *pressure towards the clause*.

We discuss the possible drawings of the clause gadget that maximize the visible boundary. There are six possible orderings for the three large disks, and all of them are symmetric. Fix one of the orderings, for example the one depicted in Figure 10. The readers do not overlap, and therefore we can maximize the visible boundary of each 1-bone independently. For the reader corresponding to direction 1, the optimal solution is to place the 1-bone with pressure towards the variable under the 3 large disks. For the remaining readers, the optimal solution is to place the 1-bone on top of the 3 large disks, and it does not matter towards where they have pressure. These drawings of the 1-bones are optimal because of their careful placement, as discussed before. We conclude that in an optimal drawing of the clause gadget there must be at least one reader with pressure towards the variable—the other readers can have arbitrary pressure.

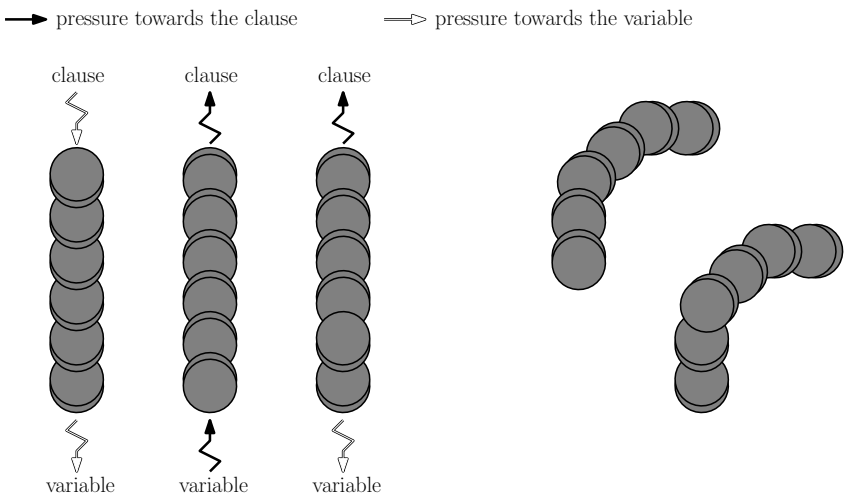


Figure 11: Channel gadgets for connecting the variables and the clauses.

Channels. We next describe the channel gadgets that connect clause and variable gadgets and represent the literals used in the clauses; see Figure 11. The gadget consists of a series of bones in the style of the spine of the variable gadget, but made with 1-bones. Note that we can make turns, as shown on the right side of the figure. A channel always uses its extreme bones to connect to a variable-reader and to a clause-reader. The key property is: In an optimal drawing of a channel gadget, it cannot happen that the clause-reader has pressure towards the variable and the variable-reader has pressure towards the clause. For all other combinations of pressures there are optimal drawings, as shown in the figure.

To decide which connections to make between variables and clauses, we look at the instance of planar 3-SAT at hand: We connect a clause-reader representing literal x with a channel to a positive variable-reader in variable x, and we connect a clause-reader representing $\overline{x}$ with a channel to a negative variable-reader in variable x. For the argument below to be correct, it is crucial that in an optimal drawing at least one reader that is connected to a variable has pressure towards that variable. Therefore we need to connect each of the six clause-readers to a variable (otherwise we could easily draw the clause optimally by putting an unused reader under the clause and the other five on top, so that all readers that are connected to variables could have pressure in any direction). Therefore we simply duplicate all literals of the clause. See Figure 12 for a combination of all the gadgets.

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.*

Proof. Consider an instance $\mathcal{I}$ of planar 3-SAT and the corresponding set of disks S. For each variable, each clause, and each channel gadget, there is a certain length of its boundary that is visible in an optimal drawing. Let G be the sum of those lengths, over all gadgets in S. Furthermore, at every connection between a channel and a reader, the last disk of the channel overlaps for a certain fraction f with the outermost disk of the reader, and at least a boundary part of length f will be covered. Let C be the number of channels in S, and let T be $G - 2fC$: Clearly T is the largest total visible boundary length one could hope to achieve. We next show that S indeed has a physically realizable drawing with visible perimeter at least T if and only if $\mathcal{I}$ is satisfiable.

We first show that if $\mathcal{I}$ is satisfiable, then S has a physically realizable drawing with visible perimeter at least T. Consider a satisfiable instance $\mathcal{I}$, and fix a Boolean assignment that

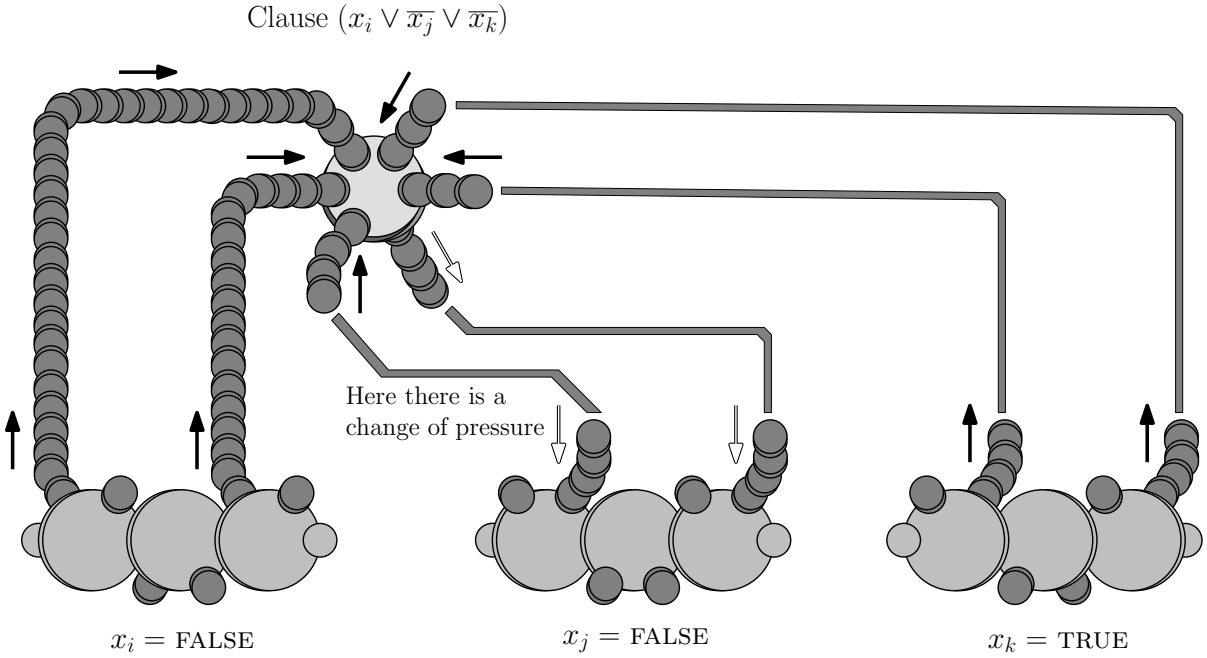


Figure 12: An example showing all the gadgets at work.

makes $\mathcal{I}$ true. For each TRUE variable in the assignment, we set the variable gadget in its TRUE state, the positive variable-readers with pressure towards the variable, and the negative variable-readers with pressure towards the clause. For each FALSE variable in the assignment, we take the symmetric drawing. In this drawing, each variable gadget gets its optimal boundary visible, and a literal is true if and only if the corresponding reader has pressure towards the variable. In each channel we consider an optimal realization that preserves the pressure, that is, both endings of the channel have pressure towards the same gadget. Since we are considering an assignment that satisfies the formula, each clause has a literal that is true, and therefore at each clause gadget a channel with pressure towards the variable arrives. This means that we can also make an optimal drawing at each clause gadget.

We now show that if S has a physically realizable drawing with visible perimeter at least T, then $\mathcal{I}$ is satisfiable. Consider a drawing of S with boundary length at least T. This means that each gadget is drawn optimally. In particular, each variable gadget must be in its TRUE or FALSE state. Now consider a clause gadget. Since it is drawn optimally, it has some clause-reader r with pressure towards the variable. Let ℓ be the clause represented by that clause-reader and let x be the variable used in ℓ for which r is a reader. Since the channel between x and r is drawn optimally, it must have pressure towards the variable at both ends. Since x is drawn optimally as well, it must therefore be in its TRUE state if r is a positive reader, or in its FALSE state if r is a negative reader. This means that clause ℓ is made true by the Boolean assignment of variables that correspond to the TRUE or FALSE states in which they are drawn. It follows that any drawing with boundary length at least T correspond directly to a Boolean assignment of variables that fulfills all clauses in $\mathcal{I}$.

For a reduction that uses integer values, and therefore a valid reduction in the RAM model, we use the same technique as in Theorem 1. Note that we can find a constant $\alpha \in (0, 1)$ with the following property: Any suboptimal drawing of a gadget (variable, clause, or channel) has a visible perimeter at least α smaller than an optimal drawing. Therefore, in the construction, it is equally hard to decide if there is a drawing of length at least T or of length strictly larger than $T - \alpha$.

Let $p_{r,r'}(d)$ be the length of the boundary of a disk D with radius r inside a disk D' with radius r', when the distance between the centers of D and D' is d. For any two intersecting disks D and D' in the construction, with radii r and r', respectively, and centers at distance d from each other, there are values $\mathit{range}_{r,r'}(d)$ and $\mathit{slope}_{r,r'}(d)$ with the following property: If we change the distance between the centers of D and D' by $\delta \leq \mathit{range}_{r,r'}(d)$, then $p_{r,r'}(d)$ changes by at most $\delta \cdot \mathit{slope}_{r,r'}(d)$. Consequently, after we have scaled our construction by a factor of at least $\sqrt{2}/\mathit{range}_{r,r'}(d)$, rounding the coordinates of the centers of the disks to the nearest integer values changes the visible boundary length of D by at most $\sqrt{2} \cdot \mathit{slope}_{r,r'}(d)$. Consider the set T of triplets (r, r', d) such that in our original construction S there are disks of radius r, r' that intersect and have their centers at distance d from each other. Let r^* be the maximum value of $\lceil \sqrt{2}/\mathit{range}_{r,r'}(d) \rceil$ and let s^* be the maximum value of $\lceil \sqrt{2} \cdot \mathit{slope}_{r,r'}(d) \rceil$ over all triplets in T. By construction T consists of a constant number of triplets, and hence r^* and s^* are constants, independent of n.

Let $|S|$ be the number of disks in S. (The value of $|S|$ is bounded by a polynomial in n, the size of the planar 3-SAT instance of size n.) Consider scaling the construction by $\lceil 1/\alpha \rceil \pi |S| s^* \max(14, r^*)$ and displacing each disk so that its center is moved to the closest point of the integer grid. In the construction, no disk intersects more than 20 disks, and therefore, the displacement changes the visible boundary of each disk by at most $20 \cdot s^*$. Since there are $|S|$ disks, the displacement changes the total visible boundary over all disks by at most $20|S|s^*$.

If the planar 3-SAT instance is satisfiable, then there is a drawing with perimeter T. After scaling and displacement, the visible boundary is at least $T_{\text{yes}} := T \cdot \lceil 1/\alpha \rceil \pi |S| s^* \max(14, r^*) - 20|S|s^*$. If the planar 3-SAT instance is not satisfiable, any drawing has perimeter at most $T - \alpha$. After scaling and displacement, the visible boundary is at most $T_{\text{no}} := (T - \alpha) \cdot \lceil 1/\alpha \rceil \pi |S| s^* \max(14, r^*) + 20|S|s^*$. Since $\alpha \cdot \lceil 1/\alpha \rceil \pi |S| s^* \max(14, r^*) > 40|S|s^*$, we have $T_{\text{yes}} > T_{\text{no}}$. Therefore, the scaled and displaced set of disks has a drawing with visible perimeter of length at least $T \cdot \lceil 1/\alpha \rceil \pi |S| s^* \max(14, r^*) - 20|S|s^*$ if and only if the planar 3-SAT instance is satisfiable. All the coordinates and radii are integers bounded by polynomials in the size of the planar 3-SAT instance, and hence this reduction can be done in polynomial time in the RAM. □

Note that the current construction might result in a physically realizable drawing that is not a stacking drawing. The reason is that a channel may impose a partial stacking order between a portion of the variable gadget and the clause gadget, and partial stacking orders arising from different channels may contradict each other. It is unclear whether the construction can be adapted so that it can be restricted to stacking drawings.

3 Algorithms

We can compute the stacking order that maximizes the minimum of the visible boundary of any symbol in polynomial time. We present the algorithms in this section. We first give a general algorithm for disks, which generalizes naturally to other convex symbols. Then we deal with special cases and squares.

Stacking general pseudo-disks in near-quadratic time. The general idea to compute a stacking order of n disks is simple: For each disk, we determine how much boundary would be seen if it were the bottommost disk. We choose the disk with the maximum value, make it the bottommost disk, and then recurse on the $n - 1$ remaining disks. To implement this greedy approach efficiently, we maintain for each disk D_i a data structure that represents all of its covered and uncovered boundary intervals. For technical reasons, we consider a disk boundary c_i to be an interval from its topmost point clockwise around. Any other disk D_j intersects c_i in

zero, one or two intervals (two if D_j contains the topmost point of c_i). The intersection points on c_i define a set of elementary intervals. In Figure 13, the elementary intervals induced by D_1, D_2, D_3, D_4 are $\alpha_1, \ldots, \alpha_9$. The data structure T_i that stores c_i is a variation of a segment tree that stores the elementary intervals in its leaves; each leaf ν stores this interval in $int(\nu)$. An internal node ν also corresponds to an interval $int(\nu)$, which is the union of elementary intervals below it in T_i. (See De Berg et al. [4] for a detailed description of segment trees.)

Every node (internal and leaf) stores the boundary length of $int(\nu)$ and a counter that stores the number of other disks that contain $int(\nu)$, but not $int(\text{parent}(\nu))$. It also stores a value *vis-int*(ν) that is the visible boundary length of $int(\nu)$ that would remain if only the disk intervals of other disks that occur in the subtree rooted at ν would hide parts of $int(\nu)$ from view. Disk intervals at ancestors of ν may still prevent any part of $int(\nu)$ from actually being visible. The root of T_i stores the total visible perimeter length of D_i—if it were placed bottommost—in *vis-int*$(\text{root}(T_i))$. The counter at the root of T_i stores the number of disks completely covering D_i.

To initialize, we construct a segment tree T_i for each disk D_i, storing the disk intervals for all disks D_j with $j \neq i$. By inspecting *vis-int*(root) and the counter for all trees $T_1, \ldots, T_n$, we determine the one with the largest boundary length if it were bottommost, and select it. When a disk D_j is chosen, we delete the disk interval of D_j from all structures T_i of disks D_i that intersect D_j and were not yet chosen. To this end, we find the canonical nodes of the disk interval of D_j in T_i. For each canonical node ν, we lower the counter. When the counter becomes 0, we also update *vis-int*(ν) by setting to the sum of the *vis-int*(..) values of the two children of ν. By the standard analysis of segment trees and tree augmentation, inserting a disk interval in T_i (at initialization) or deleting a disk interval from T_i (when the disk is chosen) takes $O(\log n)$ time. Therefore, inserting a disk's intervals in all segment trees at initialization takes $O(n \log n)$ time per disk, and the process of choosing a disk to be placed bottommost and updating all trees takes $O(n \log n)$ time per disk as well. Hence we get:

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.*

Theorem 3 also applies to pseudo-disks (sets of symbols such that the boundaries of any two symbols intersect in at most two points) of constant algebraic complexity. This includes the case when all the symbols are homothetic to a single convex polygon and are in general position.

Stacking pseudo-disks with bounded overlap in near-linear time. If no point in the plane is contained in more than C disks, where C is some constant, and the ratio in size of the largest and smallest disk is also a constant, then we can get a better bound than $O(n^2 \log n)$ as follows.

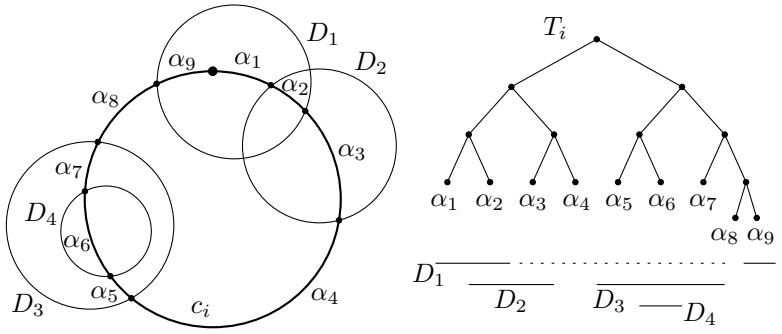


Figure 13: A disk D_i with four disks intersecting it, and the corresponding segment tree T_i.

A standard packing argument shows that in this scenario any disk intersects only a constant number of other disks. The whole arrangement of disk boundaries has complexity $O(n)$. For preprocessing, we construct the arrangement in $O(n \log n)$ time by a simple plane sweep, and for each disk, we determine the number of disks that fully contain it and store this value in a counter. For all disks not contained in any other disk we determine the visible boundary length if it were placed bottommost by walking in the arrangement along its boundary. This takes only $O(1)$ time for each disk, and $O(n)$ time overall. We store these visible boundary lengths in a priority queue.

The body of the algorithm is a loop where the next bottommost disk D_i is extracted from the priority queue, the intersecting disks are found by traversing the arrangement, their visible boundary lengths are recomputed, and their position in the priority queue is updated in $O(\log n)$ time. In $O(1)$ time we find the disks fully contained in D_i by traversing the arrangement inside D_i. For these disks we lower the counters. The ones whose counter is set to zero are put in the priority queue, after determining their visible boundary length. In this way, every selection step of a disk takes $O(\log n)$ time, so the whole algorithm takes $O(n \log n)$ time overall.

In fact we can achieve the same running time even if we assume only that no point in the plane is contained in more than a constant number of disks, and drop the assumption that the ratio in size of the smallest and largest disks is at most a constant. A disk may now intersect many more than $O(1)$ disks, but the arrangement still has $O(n)$ complexity [19], and hence all traversals to find the disks that intersect a selected disk take only $O(n)$ time in total. This implies that throughout the algorithm, only $O(n)$ visible boundary lengths are recomputed. We need to take care that this can be done efficiently for disks that intersect many other disks. To this end, we use the segment tree given above for the general algorithm, and we again obtain an $O(n \log n)$ time algorithm (note that all segment trees together have $O(n)$ leaves in this case). We get the following theorem:

Theorem 4 *Given a set of n disks in the plane such that no point is contained in more than $O(1)$ disks, a stacking order that maximizes the boundary length of the disk that is least visible can be computed in $O(n \log n)$ time.*

Like Theorem 3, Theorem 4 also applies to pseudo-disks.

Stacking unit squares in near-linear time. For unit squares we can also give an $O(n \log n)$ time algorithm without assumptions on their overlap. This is interesting because the arrangement of squares may have quadratic complexity.

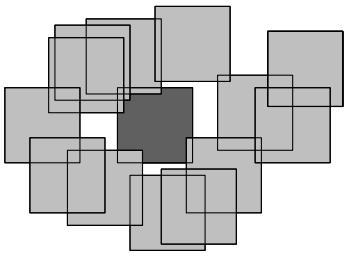


Figure 14: Deletion of a square from the union of squares.

Our algorithm for unit squares first computes the union of the squares, which has linear complexity [13] and can be computed in $O(n \log n)$ time using a plane sweep algorithm [4]. We next determine for each square the visible boundary length that it contributes to the boundary of the union. We store the squares in a priority queue ordered by this value. Note that any square has at most one visible interval on each side.

The body of the algorithm is a loop where we extract the next bottommost square S from the priority queue and update the union of squares explicitly. Up to four segments disappear, but possibly many more appear, see Figure 14. We find these by repeated ray shooting. Up to eight squares have a visible interval enlarged because they ended on a side of S. All other squares that have a change in their visible interval have a vertex exposed on the contour, or have an interval on a side exposed for the first time in the whole process. These cases can arise at most four times for each square, so the total change in intervals is $O(n)$ throughout the algorithm.

We preprocess all vertical sides of squares in a semi-dynamic data structure for horizontal ray shooting (only deletion of vertical sides needs to be supported). Similarly, we preprocess all horizontal sides of squares in a semi-dynamic data structure for vertical ray shooting. The semi-dynamic data structure of Imai and Asano [12] uses $O(\log n)$ amortized time per operation (see [9, 16] for generalizations), and therefore we can implement our algorithm to run in $O(n \log n)$ time overall.

Theorem 5 *Given a set of n unit squares in the plane, a stacking order that maximizes the visible boundary of the square that is least visible can be computed in $O(n \log n)$ time.*

Remark. Although the algorithms presented above use the absolute visible length of the boundary, they can also work with relative visible boundary length. Then we would maximize the *percentage* of the visible boundary length of the disk or square that is least visible. The adaptations needed are trivial.

4 Experiments

We have examined stacking orders based on different methods experimentally. We first describe our data sets and then the stacking methods, followed by an evaluation of their quality.

Data sets. In principle there are three different types of data sets. Either all disks have the same size, or disk sizes are taken from a small number of classes, or the disk sizes are all different and in area proportional to the relevant value. Equal size disks are uncommon because they do not show a value with locations, only an occurrence. We therefore omit such data sets from our experiments.

We used several different data sets; see Table 1. First, we took two data sets with the cities of the USA, namely the 156 and the 538 largest cities by population. The area of each disk is proportional to the population of the city. Two other data sets consist of 602 disks corresponding to earthquakes in the world. Disks are centered at the epicenters and the areas of the disks are proportional to the magnitude (scale of Richter) and to the death count [17]. Second, we used versions of the 156 cities and the earthquake magnitudes where disk sizes were classified into five different classes.

Data sets	number of disks	largest/smallest radius
City 156	156	8.4
City 538	538	21.5
Earthquake magnitude	602	1.2
Earthquake death count	602	813.0
City 156, classed	156	3.2
Earthquake mag., classed	602	2.2

Table 1: Data sets used in the experiments.

Stacking methods. Proportional symbol maps that are published in books or on the internet do not seem to follow any method consistently. Some appear to be stacked from the left to the right, others appear to be random. For maps with differently sized disks, often the stacking order is from large to small (small on top). Here we compare four different stacking methods for the four data sets with disks of arbitrary sizes and five different stacking methods for the two classed data sets. In particular, for disks of arbitrary sizes we compare the following stacking methods.

Left-to-right by center: The disk with leftmost center is put at the bottom of the stacking order, and the remaining disks are stacked recursively on top.

Left-to-right by leftmost: The disk with leftmost left extreme is put at the bottom, and the remaining disks are stacked recursively on top.

Large-to-small: The disks are stacked from bottom to top in order of non-increasing radius. If radii are distinct, then the stacking is unique.

Max-Min: The disks are stacked to maximize the visible boundary length of the disk with least visible boundary length, using the algorithm presented in Section 3.

For the classed data sets we can stack the classes from bottom to top in order of decreasing size and use one of the above methods to determine the stacking order within a class. Note that within a class all disks have equal size, so that both left-to-right methods yield the same result, while the large-to-small method would leave the order undefined. Two methods to order the disks within a class remain: left-to-right and max-min. Alternatively we could use one of the above four methods on the full set of disks—with the exception of the large-to-small method which would simply stack class by class and leave the order within a class undefined. Hence the following five stacking methods remain to be tested:

Left-to-right by center: The disk with leftmost center is put at the bottom of the stacking order, and the remaining disks are stacked recursively on top.

Left-to-right within class, classes large-to-small: Within each class the disk with leftmost center is put at the bottom of the stacking order, and the remaining disks are stacked recursively on top. The classes are then stacked bottom-to-top in order of decreasing radius.

Left-to-right by leftmost: The disk with leftmost left extreme is put at the bottom, and the remaining disks are stacked recursively on top.

Max-Min: The disks are stacked to maximize the visible boundary length of the disk with least visible boundary length, using the algorithm presented in Section 3.

Max-Min within class, classes large-to-small: Within each class we maximize the visible boundary length of the disk with least visible boundary length, using the algorithm presented in Section 3. The classes are then stacked bottom-to-top in order of decreasing radius.

All the left-to-right methods could of course also be executed from right-to-left with different results.

Evaluation. To evaluate the stacking drawings we measured the visible boundary length of the top-10 of the least visible disks and we measured the total boundary length that is visible. Disks that do not intersect any other disk were excluded from the top-10, otherwise the top-10 lists would be occupied by very small disks that are *free*, that is, disjoint from the rest and are therefore fully visible regardless of the stacking method. Observe that for sets of disks with different sizes, it may make a difference if 1 cm of the boundary of a small disk or 1 cm of the boundary of a larger disk is visible. This implies a difference in absolute visibility of a disk (in length units) and relative visibility (in percentages). Our tables include the average relative visibility of the top-10, and the average relative visibility of the whole set of disks. Note that the top-10 is selected based on absolute visibility. The average relative visibility of the top-10 shows the average percentage of the boundary that is visible, where the average is taken over the ten non-free disks with smallest absolute visibility.

Table 2 summarizes the results for the four stacking methods for the four unclassed data sets. It is clear that the max-min method performs best on the top-10 of least visible disks. The left-to-right by center method performs worst, except for the case where disks have roughly the same size (earthquake magnitudes), where the large-to-small method performs poorly. The

same observations hold for data sets that are not shown in this paper (1260 cities, 39 tsunami death counts, 33 tsunami heights). Table 3 shows the results of the five stacking methods for the two classed data sets. Again the max-min method performs best and left-to-right by center worst. The large-to-small version of the max-min method is not as good as the standard max-min method.

Besides visible boundary length, another important aspect is the visual quality of the resulting map. Since this cannot be measured, user experiments are needed to evaluate it. In this paper, we show only a few figures for comparison. Figure 15 shows (the eastern part of) the 156 largest cities of the USA with disk areas proportional to the population stacked by the four different methods (the differences can be seen most clearly in the upper right corners of the maps). The figures correspond to the top four rows of Table 2. It is noticeable that the left-to-right methods produce maps that seem 'unbalanced" or "asymmetric". A left-to-right structure is visible that has no cartographic meaning. This artifact can be perceived even more clearly on maps where the disk sizes are less different (not shown here). Finally, Figures 16 and 17 show earthquakes worldwide by magnitude and by death count, stacked by the max-min method.

The min-max method has a higher computational cost ($O(n^2 \log n)$ time) than the simple left-to-right or large-to-small methods, which require only sorting. The implementation effort is also significantly higher for the max-min method. However, it scores better than the other methods according to Tables 2 and 3, especially for the least visible disks. Furthermore, the max-min method does not have visual artifacts like the left-to-right methods, which is an advantage especially for sets of disks with little difference in size (including classed disk sizes), and on such sets of disks it clearly outperforms the large-to-small method on visible perimeter.

5 Conclusions and open problems

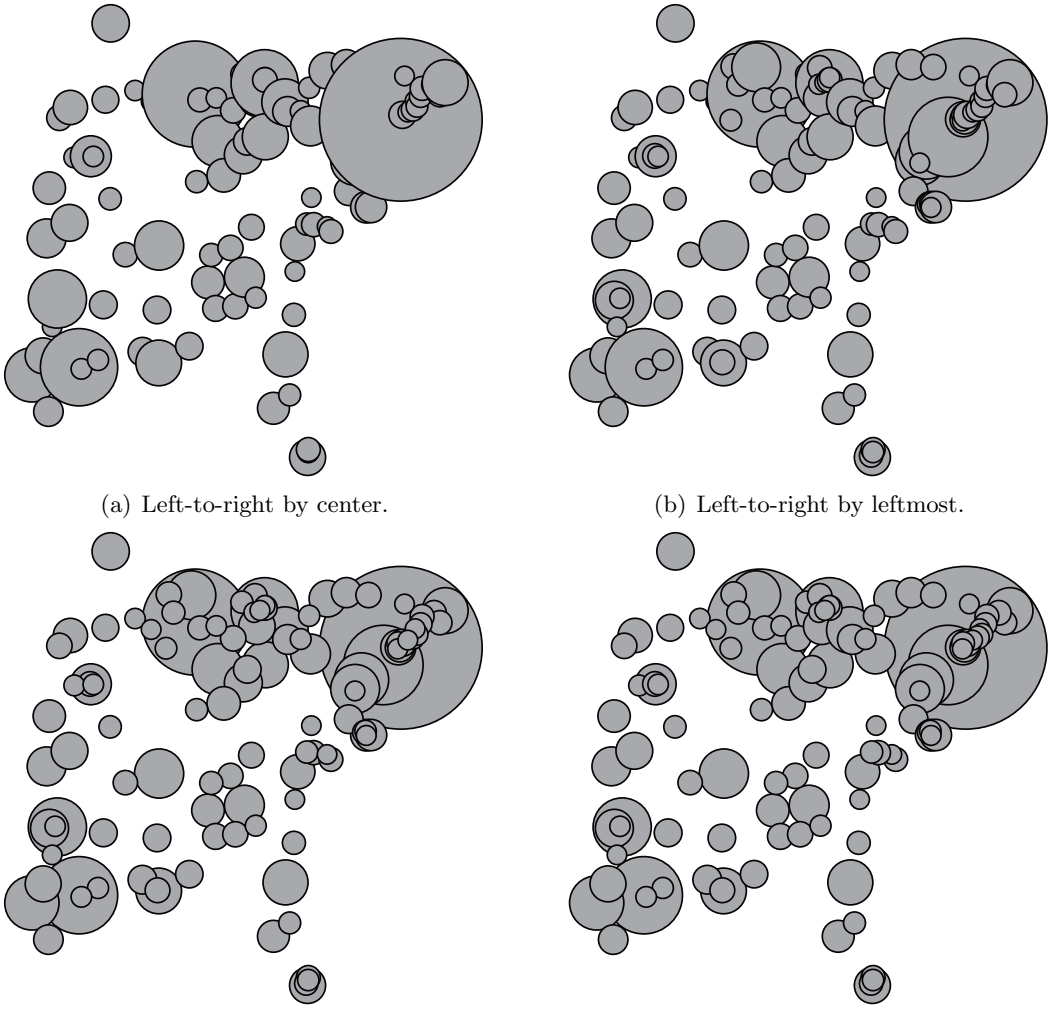
We described an algorithm that solves the Max-Min problem for stacking drawings in $O(n^2 \log n)$ time. In our experiments, comparing this algorithm with three heuristics, we found that our

Data sets		top-10 average		total boundary	
		absolute	(relative)	absolute	(relative)
City 156	Left-to-right by center	0.00	(0.00%)	1405	(57.76%)
	Left-to-right by leftmost	2.14	(21.48%)	1711	(74.84%)
	Large-to-small	2.72	(25.48%)	1730	(78.21%)
	Max-Min	4.42	(43.03%)	1759	(79.33%)
City 538	Left-to-right by center	0.00	(0.00%)	1533	(57.12%)
	Left-to-right by leftmost	0.44	(14.38%)	1815	(72.56%)
	Large-to-small	0.00	(0.00%)	1809	(73.82%)
	Max-Min	0.88	(25.91%)	1868	(76.18%)
Earthquake magnitude	Left-to-right by center	0.00	(0.00%)	14446	(44.11%)
	Left-to-right by leftmost	4.96	(9.16%)	15061	(46.11%)
	Large-to-small	0.00	(0.00%)	12126	(37.69%)
	Max-Min	12.45	(23.04%)	16608	(50.90%)
Earthquake death counts	Left-to-right by center	0.00	(0.00%)	4697	(43.74%)
	Left-to-right by leftmost	0.66	(69.93%)	8568	(87.75%)
	Large-to-small	0.78	(100.00%)	9049	(91.61%)
	Max-Min	0.78	(100.00%)	9016	(91.76%)

Table 2: Results for the data sets with disks of arbitrary sizes.

Data sets		top-10 average		total boundary	
		absolute	(relative)	absolute	(relative)
City 156	Left-to-right by center	0.00	(0.00%)	1556	(56.93%)
	Left-to-right within class, classes large-to-small	2.99	(21.13%)	1805	(72.85%)
	Left-to-right by leftmost	3.02	(25.06%)	1800	(69.55%)
	Max-Min	5.30	(43.02%)	1844	(74.02%)
	Max-Min within class, classes large-to-small	4.80	(36.16%)	1829	(73.89%)
Earthquake magnitude	Left-to-right by center	0.00	(0.00%)	12123	(45.06%)
	Left-to-right within class, classes large-to-small	1.38	(3.03%)	13286	(55.81%)
	Left-to-right by leftmost	3.38	(10.19%)	13717	(54.72%)
	Max-Min	10.78	(37.26%)	14429	(59.31%)
	Max-Min within class, classes large-to-small	6.41	(14.31%)	13915	(58.09%)

Table 3: Results for the classed data sets.



(a) Left-to-right by center.

(b) Left-to-right by leftmost.

(c) Large-to-small.

(d) Max-Min.

Figure 15: USA, 156 biggest cities, showing only the right half of the map.

method performed best on the test data considered. However, we did not experiment with methods that compute physically realizable drawings, and it is unclear how they would perform

in comparison with our methods for stacking drawings. Solving the Max-Min problem (or the Max-Total problem) for physically realizable drawings is NP-hard, and developing good heuristics for such drawings is not trivial.

Among the open problems that remain are the computation of optimal Max-Min stacking drawings in $o(n^2 \log n)$ time, the computation of optimal Max-Total stacking drawings (or approximations thereof) in polynomial time, and the development of approximation algorithms for physically realizable drawings.

Acknowledgement. The authors thank Christian Vossers for implementing the methods and running the experiments.

References

[1] P. K. Agarwal and S. Suri. Surface approximation and geometric partitions. *SIAM Journal on Computing*, 27:1016–1035, 1998.

[2] S. Cabello, E. D. Demaine, and G. Rote. Planar embeddings of graphs with specified edge lengths. In *Proceedings of the 11th International Symposium on Graph Drawing. LNCS*, vol 2912, pp. 283–294, 2004.

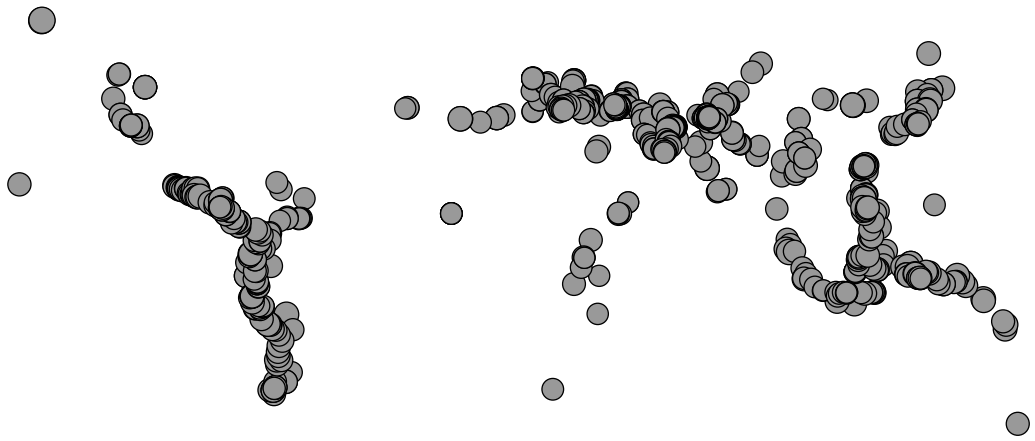


Figure 16: Earthquake magnitudes (Richter scale) stacked by the Max-Min method.

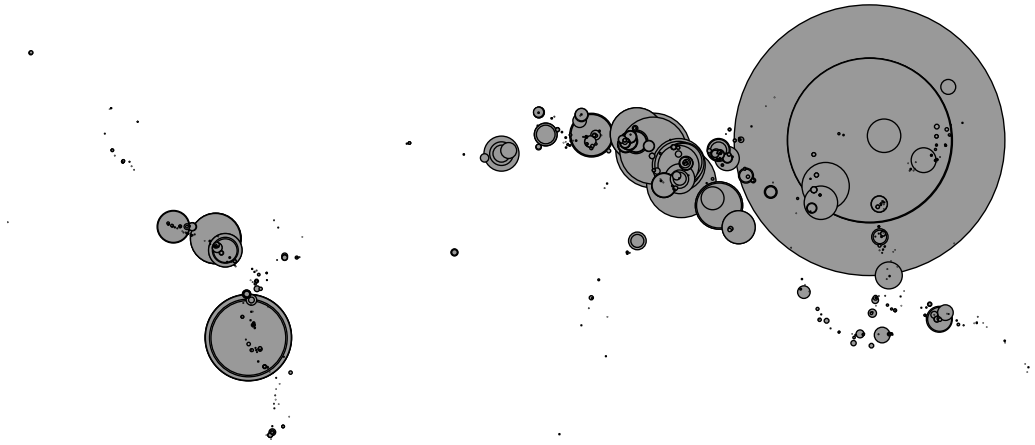


Figure 17: Earthquake death counts stacked by the Max-Min method.

[3] K. C. Clarke. *Analytical and Computer Cartography*. Prentice Hall, 2nd edition, 1995.

[4] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd edition, 2008.

[5] E. D. Demaine, J. S. B. Mitchell, and J. O'Rourke. The open problems project. Problem 33. `http://maven.smith.edu/ orourke/TOPP/P33.html`.

[6] B. Dent. *Cartography - thematic map design*. McGraw-Hill, 5th edition, 1999.

[7] L. Fortnow. Computational Complexity Blog. Post of Friday, February 14, 2003. `http://weblog.fortnow.com/archive/2003_02_01_archive.html`.

[8] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*, 12(3):133–137, 1981.

[9] Y. Giora and H. Kaplan. Optimal dynamic vertical ray shooting in rectilinear planar subdivisions. In *Proceedings of the 18th ACM-SIAM Symposium on Discrete Algorithms*, pp. 19-28, 2007.

[10] T. Griffin. The importance of visual contrast for graduated circles. *Cartography*, 19(1):21–30, 1990.

[11] R. E. Groop and D. Cole. Overlapping graduated circles: Magnitude estimation and method of portrayal. *Canadian Cartographer*, 15(2):114–122, 1978.

[12] H. Imai and T. Asano. Dynamic orthogonal segment intersection search. *Journal of Algorithms*, 8:1–18, 1987.

[13] K. Kedem, R. Livne, J. Pach, and M. Sharir. On the union of Jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete & Computational Geometry*, 1:59–71, 1986.

[14] D. E. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM Journal on Discrete Mathematics*, 5:422–427, 1992.

[15] D. Lichtenstein. Planar formulae and their uses. *SIAM Journal on Computing*, 11(2):329–343, 1982.

[16] K. Mehlhorn and S. Näher. Dynamic Fractional Cascading. *Algorithmica*, 5:215–241, 1990.

[17] NOAA Satellite and Information Service. National geophysical data center, 2005. `http://www.ngdc.noaa.gov`.

[18] Queensland University Advanced Centre for Earthquake Studies. `http://www.quakes.uq.edu.au`.

[19] M. Sharir. On k-sets in arrangements of curves and surfaces. *Discrete & Computational Geometry*, 6:593–613, 1991.

[20] T. A. Slocum, R. B. McMaster, F. C. Kessler, and H. H. Howard. *Thematic Cartography and Geographic Visualization*. Prentice Hall, 2nd edition, 2003.