Optimizing Symbol Visibility Through Displacement

Bernd Gärtner ✉
Department of Computer Science, ETH Zürich, Switzerland

Vishwas Kalani ✉
Department of Computer Science and Engineering, I.I.T. Delhi, India

Meghana M. Reddy[1] ✉ 🆔
Department of Computer Science, ETH Zürich, Switzerland

Wouter Meulemans ✉
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Bettina Speckmann ✉ 🆔
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Miloš Stojaković ✉ 🆔
Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Serbia

—— **Abstract** ——

In information visualization, the position of symbols often encodes associated data values. When visualizing data elements with both a numerical and a categorical dimension, positioning in the categorical axis admits some flexibility. This flexibility can be exploited to reduce symbol overlap, and thereby increase legibility. In this paper we initialize the algorithmic study of optimizing symbol legibility via a limited displacement of the symbols.

Specifically, we consider unit square symbols that need to be placed at specified y-coordinates. We optimize the drawing order of the symbols as well as their x-displacement, constrained within a rectangular container, to maximize the minimum visible perimeter over all squares. If the container has width and height at most 2, there is a point that stabs all squares. In this case, we prove that a staircase layout is arbitrarily close to optimality and can be computed in $O(n \log n)$ time. If the width is at most 2, there is a vertical line that stabs all squares, and in this case, we give a 2-approximation algorithm (assuming fixed container height) that runs in $O(n \log n)$ time. As a minimum visible perimeter of 2 is always trivially achievable, we measure this approximation with respect to the visible perimeter exceeding 2. We show that, despite its simplicity, the algorithm gives asymptotically optimal results for certain instances.

---

[1] The third author's full last name consists of two words and is *Mallik Reddy*. However, she consistently refers to herself with the first word of her last name being abbreviated.

1 Introduction

When communicating information visually, the position of symbols is an important visual channel to encode properties of the data. For example, in a scatter plot that visualizes age versus income of a given population, each data item (a person in the population) is visualized with a symbol (commonly a square, a cross, or a circle) which is placed at an x-coordinate that corresponds to their age and a y-coordinate that corresponds to their income. Hence persons with similar values are placed in close proximity, which allows the user to visually detect patterns. Another example from cartography are so-called proportional symbols maps which visualize numerical data associated with point locations by placing a scaled symbol (typically an opaque disc or square) at the corresponding point on the map. The size of the symbol is proportional to the data value of its location, such as the magnitude of an earthquake. The density and size of the symbols again supports visual pattern detection.

In both examples above, the position of the symbol is fixed and cannot be changed without severely distorting the information it encodes. In other settings, such as map labeling, the position of a symbol is not completely fixed, instead the symbol (the label) needs to be placed in contact with a particular point on the map. There are infinitely many potential placements for the symbol, but all must contain the same fixed point somewhere on its boundary. In this paper we consider a related symbol placement problem, which is motivated by the visualization of numerical data with associated categories: the age of employees within a certain division of a company or the page rank of tweets that exhibit a certain sentiment (positive, negative, neutral) on a topic such as vaccinations. Such data can be visualized in *categorical strips* of fixed width w, restricting the symbols to lie in the strip, and placing the symbols on a y-coordinate according to their numerical values (see Figure 1 for an example using twitter data). There are again infinitely many potential placements per symbol, but all placements are restricted to share the same y-coordinate.

If the positions of symbols are fixed or restricted, then close symbols will overlap, reducing the visible part of – or even fully obscuring – other symbols. Correspondingly, there is an ample body of work on optimizing the visibility of symbols under placement restrictions. The algorithmic literature considers a couple of variants. First of all, we either display all symbols or only a subset. For symbol maps and our categorical strips we always have to display all symbols, since otherwise not all data is visualized. For map labeling one usually chooses a subset of the labels which can be placed without any overlap; the corresponding optimization problems attempt to maximize the number of these labels while also taking priorities (such as city sizes) into account. If overlap between symbols might be unavoidable, visibility is optimized by either maximizing the minimum perimeter of the symbol that has least visibility or maximising the total visible perimeter. If the positions of the symbols are completely fixed, then the only choice we can make is the drawing order of the symbols.

In this paper we study the novel algorithmic question of how to optimize the visibility of a given set of symbols, all of which must be drawn, when we may choose their drawing order and their x-coordinate, given a set of fixed (and distinct) y-coordinates for each symbol. We measure the visibility of the result via the minimal visibility perimeter over all symbols [3]. Figure 1 shows that our algorithms do indeed greatly improve the visible perimeter and thereby give the viewer a more accurate impression of the data. Note that our theoretical results hold only for square symbols, but, as evidenced by Figure 1, the algorithms we propose readily extend to rectangular symbols. Proving similar bounds for more general symbol shapes is a challenging open problem.

Figure 1 A pro and con vaccination twitter discussion from 2018, capturing 823 accounts. Accounts are placed in a categorical strip according to most frequent sentiment expressed. Color indicates if account mostly mentions accounts with a similar opinion (blue) or with a different opinion (red). Top: random jittering, bottom: our approximation algorithm described in Theorem 7. Random jittering hides many details, such as pro-vaccination accounts that mention predominantly anti-vaccination accounts (red boundaries). Many accounts at the bottom sent the same number of tweets and hence share a y-coordinate, which inevitably covers most of their horizontal edges.

Contributions and organization. In this paper we initiate the algorithmic study of optimizing symbol visibility through displacement. Specifically, we focus on unit square symbols, that may be shifted horizontally while remaining in a strip of width 2 (their categorical strip). In Section 2 we introduce our notation and make some initial observations. Most notably: the visible perimeter behaves non-continuously when squares are placed on the same x-coordinate. Hence the optimal visible perimeter is a supremum that cannot always be reached. In Section 3 we study the special case that the strip has height at most 2. In this scenario all squares are stabbed by a point. We first establish several useful geometric properties of so-called reasonable layouts – arrangements of the input squares which meet

certain lower bound conditions – and then use these properties to prove that a simple $O(n \log n)$ algorithm suffices to compute a layout of the squares whose visible perimeter is arbitrarily close to the supremum.

In Section 4 we then study the general case of strips of arbitrary height (but still width 2). Here all squares are stabbed by a line. We leverage our previous result to obtain an $O(n \log n)$-time approximation algorithm. This approximation is with respect to the so-called gap – the visible perimeter minus two – since a minimal visible perimeter of 2 is trivially obtained for any instance. Furthermore, if the y-coordinates are uniformly distributed, then we can show that a specific layout – the zigzag layout – is asymptotically optimal. We close with a discussion of various avenues for future work, both towards the practical applicability of our results in visualization systems and towards more theoretical results in other settings, including different visibility definitions and other symbol shapes.

Related work. The questions we study in this paper combine various aspects and restrictions of well-known placement problems in the algorithmic (geo-)visualization literature in a novel way. Most closely related to our work are the two papers by Cabello et al. [3] and by Nivasch et al. [15] that consider perimeter problems for sets of differently sized symbols at fixed positions in the plane. Specifically, they consider the problems of maximizing the minimum perimeter of the symbol that has least visibility (as we do in this paper) or maximizing the total visible perimeter. Since the locations of the symbols are fixed, the algorithmic problem reduces to finding the optimal (not necessarily stacking) order of the symbols. This contrasts with our work where a limited form of displacement is allowed.

Labeling cartographic maps, where a subset of the labels are chosen such that they can be placed without any overlap, and the corresponding optimization problems are computationally complex in various settings [8], as they relate to maximum independent set. Constrained displacement of labels is often also allowed, and various of such placement models have been studied algorithmically [2, 4, 16, 17, 21]. The goal is always to place as many labels as possible without overlap. Displacing labels to the boundary of a map has attracted considerable algorithmic attention under different models as well, see [1] for a survey. However, such practice relies on leader lines to connect labels to the points being labeled; making it harder to identify data patterns and thus less suitable for visualizing data.

There is ample work on using symbol displacement to eliminate all symbol overlap, as it has various applications in visualization, including visualizing disjoint glyphs in geovisualization [11, 13, 20], removing overlap between vertices in graph drawing [6, 12], positioning nonspatial data [10, 18] and computing Dorling and Demer's cartograms [5, 14]. Such overlap-removal problems are NP-hard in many settings [7, 19], though efficiently solvable in some specific settings [13, 14]. However, eliminating all overlap may require considerable displacement, thereby distorting the view of the data. Our goal is not to eliminate all overlap, but to use both a limited displacement and a suitable drawing order to maximize visibility.

In the visualization literature, offsetting graphical symbols to improve visibility is known as jittering [22]. Often, jittering is done randomly, though in context of dense plots, arising, for example, from dimensionality reduction, more complex algorithms have been engineered for this task; see e.g. [9] for a recent method. However, such approaches are often heuristic in nature, without provable quality guarantees.

2   Preliminaries

Our input is a sequence of distinct y-coordinates $\mathbf{y} = (y_1, y_2, \ldots, y_n)$. We want to find x-coordinates $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, determining unit squares $s_1, s_2, \ldots, s_n$, where s_i has centroid (x_i, y_i). We also want to find a stacking order, so that the minimum visible perimeter among all squares is maximized.

More specifically, we are given a *strip T* of width $w > 1$ and height $h > 1$, where we assume that $T = [0, w] \times [0, h]$, and that $\frac{1}{2} \le y_1 < y_2 < \ldots < y_n \le h - \frac{1}{2}$. Thus, in terms of the y-coordinates, we can speak about the *highest*, or the *lowest*, among any set of squares. We require $x_i \in [\frac{1}{2}, w - \frac{1}{2}]$ for all i, so that all squares are in the strip. We say that s_i is *left* (*right*) of s_j if $x_i < x_j$ ($x_i > x_j$). This determines a *leftmost* and a *rightmost* square among any set of squares (ties handled as needed).

A *stacking order* is a total order $\prec$ among the squares. If $i \prec j$, we say that s_j is *in front of s_i*, and s_i is *behind s_j*. The *bottom* and *top square* are the first and last squares in the order. The pair $(\mathbf{x}, \prec)$ is a *layout* for the instance $(w, h, \mathbf{y})$.

The *visible perimeter* of a square in a layout is the total length of all its *visible* boundary, where a point on the boundary is visible if any other square t containing it is behind s. The top square has visible perimeter 4. The visible boundary is made up of (horizontal or vertical) *visible edges*. Note that each side of s can host at most one visible edge, due to all squares having the same size.

The *gap* of a square in a layout is its visible perimeter minus 2. If this is non-positive, we say that the square has no gap. The gap of a layout is the minimum of the gaps of all squares. This definition is motivated by the fact that we can always achieve a positive gap by suitable "standard" layouts which we introduce below. Ideally, we want to find an optimal layout, one that has the maximum gap among all layouts. However, this may not exist: one can easily construct instances where the only candidates for optimal layouts have duplicate x-coordinates, but no such layouts can actually be optimal, refer to Figure 2 for one such instance. Therefore, we take the supremum gap over all layouts as the benchmark which we want to approximate as closely as possible.

The *bounding box* of a collection of squares is the inclusion-minimal axis-parallel box containing all the squares.

We call a layout a *staircase* if both $\mathbf{x}$ and $\prec$ are monotone:

- $x_1 \le x_2 \le \cdots \le x_n$ ("facing right"), or $x_1 \ge x_2 \ge \cdots \ge x_n$ ("facing left"); and
- $s_1 \prec s_2 \prec \cdots \prec s_n$ ("facing up"), or $s_1 \succ s_2 \succ \cdots \succ s_n$ ("facing down").

Hence, there are 4 types of staircases; the one in Figure 3 (left) is facing right and up.
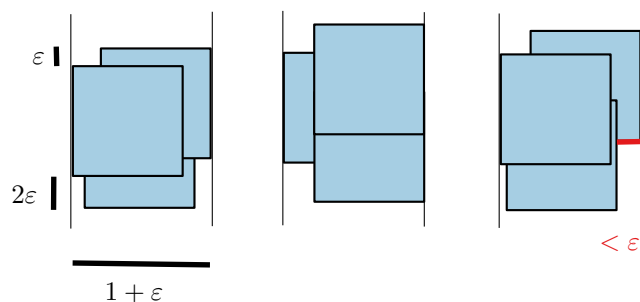
Figure 2 An instance with $y_2 - y_1 = 2\varepsilon, y_3 - y_2 = \epsilon$ and a strip of width $w = 1 + \varepsilon$, where $\varepsilon \le 0.2$. From only looking at the two highest squares s_2, s_3, we see that the gap of every layout is bounded from above by 2ε, and to achieve this gap, s_2 and s_3 together need to span the full strip width. Given this, s_1 cannot be the bottom square, since then it has visible perimeter at most $1 + 5\varepsilon \le 2$ and hence no gap *(left)*; s_2 cannot be the bottom square, either, as this would limit its visible perimeter to at most $1 + 2\varepsilon$ *(middle)*. So s_3 has to be the bottom square *(right)*. Since $x_1 = x_2$ would give the bottom square among s_1, s_2 a visible perimeter of at most $1 + 4\varepsilon$, we need to have $x_1 \ne x_2$, but this means that s_3 has gap less than 2ε. Hence, a gap of 2ε is not achievable, but any smaller gap is (by the right layout, as $x_1 - x_2 \to 0$).
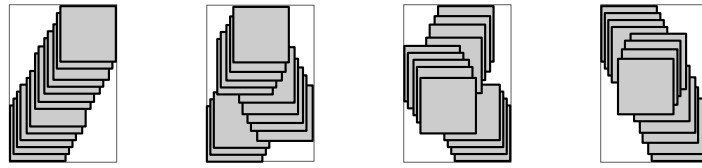
Figure 3 A staircase (left), three generalized staircases (right).

We call a layout a *generalized staircase* if each square s_i lies in one of the four corners of the bounding box of s_i and all squares in front of it. In a standard staircase, this corner is the same for all squares (i.e., the lower left corner for a staircase facing right and up).

▶ **Observation 1.** *For every instance, there is a staircase with positive gap.*

Proof. We build a staircase facing right and up as in Figure 3 (left). Then the left and lower sides of each square are completely visible, as well as parts of its right and upper side. This yields a positive gap. ◀

▶ **Definition 2.** *A layout is* reasonable *if it has positive gap, and it is ε-reasonable if it has gap larger than $\varepsilon > 0$.*

3   Squares stabbed by a point

Throughout this section, we fix a strip width $w \leq 2$ and a strip height $h \leq 2$. In this case, all squares are stabbed by a single point. Subsection 3.1 proves a (tight) upper bound on the gap of every layout, while Subsection 3.2 shows that a staircase layout of gap arbitrarily close to the supremum can efficiently be computed.

3.1   Reasonable layouts

We start with a crucial structural result about reasonable layouts in the case $w, h \leq 2$.

▶ **Proposition 3.** *In a reasonable layout, the bottom square s is not contained in the bounding box of the other squares.*

Proof. Assume otherwise and consider the two squares s_ℓ and s_r defining the left and right sides of the bounding box. If one of them is above s and the other one below, the situation is as in Figure 4 (left). Since s_ℓ and s_r overlap in both x- and y-coordinate, s has horizontal and vertical visible edges of total length at most 1 each. Thus, the visible perimeter of s is at most 2. In the other case, s_ℓ and s_r are w.l.o.g. both above s as in Figure 4 (right). Then we consider the square s_d defining the bottom side of the bounding box; w.l.o.g. s_d is left of s. In this case, s_r and s_d prove that s has no gap. ◀

▶ **Lemma 4.** *Every layout $\mathcal{A}$ of n squares has gap at most $\frac{w+h-2}{n-1}$, for $n \geq 2$.*

Proof. If $\mathcal{A}$ is unreasonable, there is nothing to prove. Otherwise, let $t_1, t_2, \ldots, t_n$ be the sequence of squares in stacking order, i.e. $t_1 \prec t_2 \prec \cdots \prec t_n$, and let the bounding box of $t_i, t_{i+1}, \ldots, t_n$ be denoted by τ_i for $1 \leq i \leq n$. Since $\mathcal{A}$ is reasonable, t_i "sticks out" of τ_{i+1} (i.e. it is not contained in it) by Proposition 3. There are two cases: t_i is a "corner square" (Figure 5 left), or a "side square" (Figure 5 middle and right). Let ΔX_i and ΔY_i quantify by how much t_i sticks out, horizontally and vertically. For a side square, one of those numbers is 0.
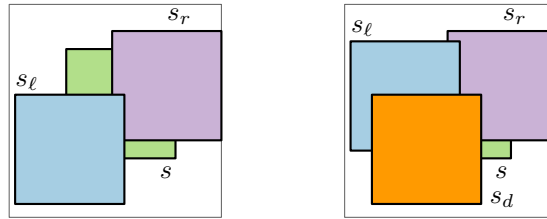
Figure 4 The visible perimeter of bottom square s is at most two if it is contained in the bounding box of the other squares.

For a corner square, the two sides of t_i incident to the corner contribute visible perimeter 2, meaning that the gap of the square is $\Delta X_i + \Delta Y_i$. For a horizontal side square as in Figure 5 (middle), the visible perimeter is at most $2 + \Delta Y_i$, and for a vertical side square (right), it is at most $2 + \Delta X_i$. In both cases, $\Delta X_i + \Delta Y_i$ is also an upper bound for the gap of the square.

This means that, for $1 \le i < n$, the intervals corresponding to ΔX_i (as well as those corresponding to ΔY_i) span disjoint x-intervals (or y-intervals) that are also disjoint from the two intervals (one in each coordinate) of length one that is spanned by the top square. Hence,

$$\sum_{i=1}^{n-1}(\Delta X_i + \Delta Y_i) \le (w-1) + (h-1) = w + h - 2.$$

It follows that there is some t_i with gap at most $\Delta X_i + \Delta Y_i \le \frac{w+h-2}{n-1}$. ◀

This upper bound on the gap is easily seen to be tight.

▶ **Observation 5.** *There are instances of n squares for which a staircase has gap $\frac{w+h-2}{n-1}$.*

Proof. We consider the uniformly spaced instance $(y_i = \frac{1}{2} + (h-1)\frac{i-1}{n-1})$. Choosing $x_i = \frac{1}{2} + (w-1)\frac{i-1}{n-1}$ and $s_1 \prec s_2 \cdots \prec s_n$ leads to a staircase with $\Delta X_i = \frac{w-1}{n-1}$ and $\Delta Y_i = \frac{h-1}{n-1}$ for $1 \le i < n$, and hence the gap is $\frac{w+h-2}{n-1}$. ◀
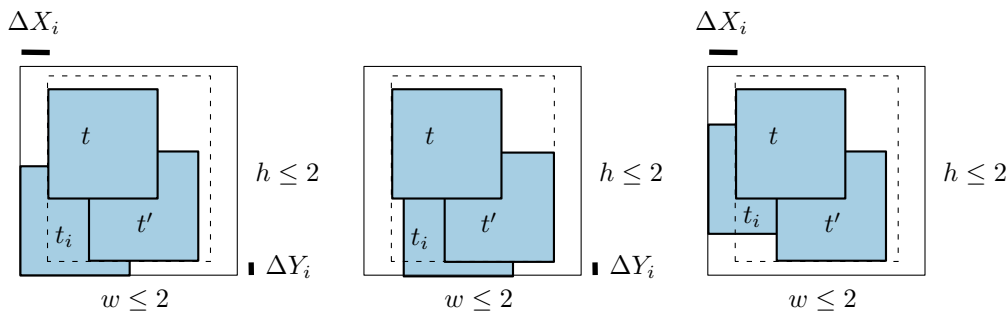
Figure 5 Proof of Lemma 4.

3.2   Computing staircases with gap arbitrarily close to the supremum

We next prove that for every reasonable layout with gap g, there is a staircase with a gap at least $g - \delta$, for any $\delta > 0$. Moreover, with $\gamma^\star$ being the supremum gap over all layouts, a staircase of gap $\gamma^\star - \delta$ can be efficiently computed.

For this, we first look at staircases in more detail. Consider a staircase of n squares, facing right and up, with centroids (x_i, y_i), $1 \leq i \leq n$. We define $\Delta y_i = y_{i+1} - y_i > 0$ and $\Delta x_i = x_{i+1} - x_i \geq 0$, for $1 \leq i \leq n - 1$. If $\Delta x_i > 0$, the left and lower sides of s_i are fully visible, and the gap of s_i is $\Delta y_i + \Delta x_i$; the top square s_n has gap 2. If all Δx_i are positive, the staircase is called *proper*.

Now consider the problem of finding such a proper staircase of large gap. For this, the Δy_i are fixed, but $x_1 < x_2 < \cdots < x_n$ can be chosen freely, meaning that the values Δx_i can be any positive numbers satisfying $\sum_{i=1}^{n-1} \Delta x_i \leq w - 1$. We want to maximize the gap $\min_{i=1}^{n-1}(\Delta y_i + \Delta x_i)$ subject to the previously mentioned constraints. Due to the strict inequalities on the Δx_i's, the maximum may not exist (as pointed out in Section 2). But allowing $\Delta x_i \geq 0$, the maximum is attained by the solution of a linear program:

$$
\begin{aligned}
\text{maximize } & g \\
\text{subject to } \quad & \Delta x_i + \Delta y_i \;\geq\; g, \quad i = 1, \ldots, n - 1 \\
& \textstyle\sum_{i=1}^{n-1} \Delta x_i \;\leq\; w - 1 \\
& \Delta x_i \;\geq\; 0, \quad i = 1, \ldots, n - 1.
\end{aligned}
\tag{1}
$$

Now we are prepared for the main result of this section which also implies that the optimal solution $g^\star$ of this linear program equals $\gamma^\star$, the supremum gap over all layouts.

▶ **Lemma 6.** *Let $\mathcal{A}$ be a reasonable layout with gap g. There exists a feasible solution of the linear program (1) with value at least g. Moreover, for every $\delta > 0$, there exists a proper staircase $\mathcal{A}'$ with gap at least $g - \delta$.*

Proof. As in the proof of Lemma 4, we consider the ΔX_i and ΔY_i, $i = 1, \ldots, n - 1$, quantifying by how much the i-th square in the stacking order sticks out of the bounding box of the squares in front of it, horizontally and vertically. See Figure 6 (top left part) for an illustration, where the ΔX_i and ΔY_i values are visualized as rectangle areas. Each rectangle is spanned by a unit side and an *interval* corresponding to the value. For example, ΔX_4, the rectangle with the orange boundary, has the interval between the left sides of t_4 and t_5.

The ΔY_i rectangles are further subdivided into *blocks* whose intervals are gaps between vertically adjacent squares. For example, ΔY_4 is made of two blocks. The interval of the orange-black block is the vertical gap between the lower sides of t_4 (orange) and t_1 (black), while the interval of the black-green block is the vertical gap between the lower sides of t_1 (black) and t_5 (green). Thus, each block interval is of the form $\Delta y_j = y_{j+1} - y_j$.

As argued in the proof of Lemma 4, the ΔX_i and ΔY_i satisfy the constraints of the linear program (1), with g being the gap of $\mathcal{A}$.

We will perform discrete steps that gradually turn the ΔY_i into the prescribed $\Delta y_i = y_{i+1} - y_i$, while changing the ΔX_i into suitable Δx_i. If we can maintain the constraints of (1) throughout, we will arrive at a feasible solution of (1) – that we can interpret as a staircase – with value at least g. From this, we can construct a proper staircase with gap at least $g - \delta$, by slightly redistributing the Δx_i to make all of them positive.

We point out that the $\Delta X_i, \Delta Y_i$ are sorted by stacking order; our stepwise process will first result in values $\Delta X_i', \Delta Y_i'$ such that the $\Delta Y_i'$ are a permutation of the Δy_i. This means, we still have to sort the values accordingly before we can interpret the solution of (1) as a staircase. As the linear program is agnostic to permutations, this does not change the gap.
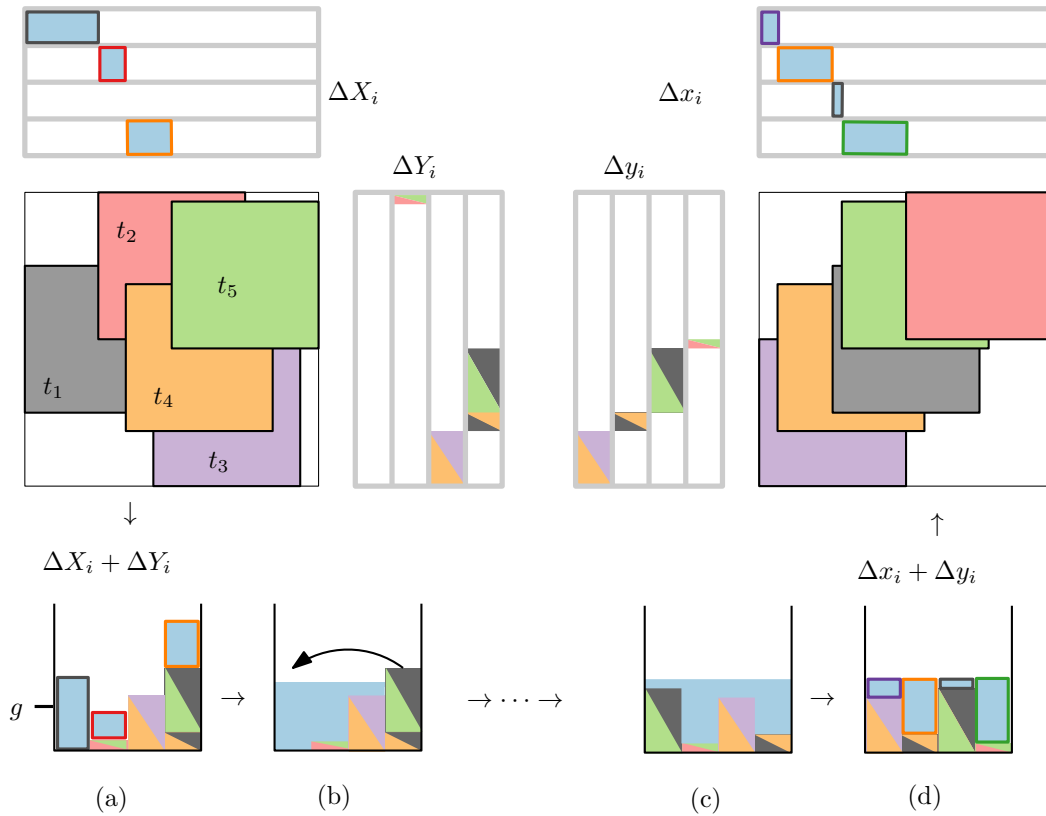
Figure 6 The proof of Lemma 6.

If the ΔY_i are already a permutation of the Δy_i, we are done after sorting them (see the end of the proof below). This is the case if and only if each ΔY_i consists of exactly one block.

But in general, some ΔY_i may have more than one block, or no block at all. In the example in Figure 6, we have $\Delta Y_4 = y_4 - y_2$ consisting of two blocks with intervals between t_4 (orange) and t_1 (black), and between t_1 (black) and t_5 (green). ΔY_1 in turn has no blocks, as t_1 does not stick out vertically.

All the ΔY_i together use all the $n - 1$ blocks. Indeed, the bounding box of the squares in front of t_i is disjoint from whatever sticks out of it, and the last bounding box only contains the top square.

Now we repeatedly move blocks from rectangles with at least two blocks to rectangles with no block. We can visually analyze this as follows: think of a basin that initially holds the ΔY_i rectangles, $i = 1, \ldots, n - 1$, as in part (a) of Figure 6. For each i, we pour ΔX_i units of water into the basin, which corresponds to the area of the rectangles with colored boundary in part (a) of Figure 6. As $\min_i(\Delta X_i + \Delta Y_i) \geq g$, the water will settle at some level $\geq g$; see part (b) of the figure. Moving a block to a "free slot" will submerge it further, and this can only increase the water level; see step (b)-(c).

In the end, we have one block $\Delta y_j = y_{j+1} - y_j$ per slot, and sorting the slots by index as in part (d) yields rectangles $\Delta y_1, \ldots, \Delta y_{n-1}$, with columns $\Delta x_1, \ldots \Delta x_{n-1}$ of water above them, such that $\min(\Delta x_i + \Delta y_i) \geq g$. In general, some Δy_i's can still be above the water level in which case the corresponding Δx_i is 0. This is our desired solution of (1) from which we can in turn build a staircase with the prescribed x- and y-gaps (upper right part of the figure). ◀
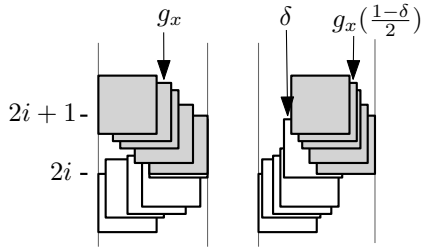
Figure 7 Squeezing staircase layouts.
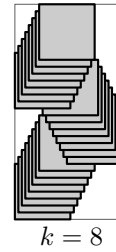
Figure 8 Arranging n uniformly spaced squares in a zigzag layout.

We remark that the linear program (1) can be efficiently solved in $O(n \log n)$ time, employing the water analogy. After sorting the Δy_i rectangles, and assuming that the water currently rises to the top of one of them, it is easy to compute in $O(1)$ time the amount of additional water required to reach the top of the next higher rectangle. Indeed, in this range, the water level is a linear function of the amount of additional water. If reaching the top of the next higher rectangle would need more water than our total budget of $w - 1$ allows, we arrive at the optimal level $g^\star$ before.

4   Squares stabbed by a vertical line

Throughout this section, we consider a strip of width $w \leq 2$ and arbitrary height $h > 1$, with n squares of fixed y-coordinates $\frac{1}{2} \leq y_1 < y_2 < \cdots < y_n \leq h - \frac{1}{2}$. Let $1/k = (h-1)/(n-1)$ be the (maximal) average y-distance between adjacent centroids in the y-order. We first show that we can asymptotically approximate the supremum gap up to a factor of 2. More precisely, as the strip remains fixed and $n \to \infty$, we have $1/k \to 0$ and thus approach a factor of 2 using Theorem 7 below. We still present our results in terms of k to make it clear what happens if the strip height h grows with n.

▶ **Theorem 7.** *Let $\gamma^\star$ be the supremum gap over all layouts. In time $O(n \log n)$, we can construct a layout with gap at least $\gamma^\star(\frac{1}{2} - O(\frac{1}{k}))$. We refer to this procedure as the* squeezing *algorithm.*

Proof. We partition the squares into buckets $1, \ldots, \lceil h \rceil$, where bucket i contains the squares j such that y_j rounds to i (we round up in case of a tie). The squares within each bucket are in a strip of height 2, and by Section 3, a (staircase) solution of gap arbitrarily close to the supremum can efficiently be found, in time $O(\ell \log \ell)$ per bucket, where ℓ is the number of squares in that bucket. Hence, the total time required is $O(n \log n)$.

The smallest bucket gap δ is (up to arbitrarily small error) an upper bound for $\gamma^\star$, as each layout contains a sublayout for the squares in this worst bucket. We also note that $\delta = O(\frac{1}{k})$, since there must be a bucket with $\Omega(k)$ squares to which Lemma 4 applies.

In $O(n)$ time, we now construct a layout for all squares, of gap roughly $\frac{\delta}{2}$, to prove the statement. To do so, we "squeeze" the layouts in individual buckets appropriately. We assume w.l.o.g. that the even bucket staircases are facing right and up, while the odd ones are facing left and up, as in Figure 7 (left).

Multiplying all x-gaps by $\frac{1-\delta}{2}$ while keeping the even staircases aligned left and the odd ones aligned right, see Figure 7 (left), leads to a layout where even staircase squares have $x \leq 1 - \frac{\delta}{2}$, and odd ones have $x \geq 1 + \frac{\delta}{2}$. Each non-top square of each bucket still has gap $g_y + g_x \frac{1-\delta}{2}$ where g_x, g_y are the previous x-gap and y-gap in the bucket solution, and

$g = g_x + g_y \geq \delta$ is the previous gap. It follows that the new gap is at least $\delta \frac{1-\delta}{2}$. The top squares of each bucket have x-gap (and hence total gap) at least δ, by construction. The resulting layout has therefore gap at least $\delta \frac{1-\delta}{2}$. Since $\gamma^\star \gtrapprox \delta = O(1/k)$, the bound follows. ◀

It is natural to ask whether squeezing the staircase layouts of individual buckets is the best we can do. For general y_i, we do not know the answer, but if the y_i are uniformly spaced, we can indeed prove that this procedure yields an asymptotically optimal gap.

Uniform spacing. For the rest of the section, we assume that $y_{i+1} - y_i = \frac{1}{k}$ for $1 \leq i < n$. In this case, the squeezing algorithm from Theorem 7 essentially produces the *zigzag* layout (see Figure 8).

▶ **Lemma 8.** *The zigzag layout has gap* $\frac{1}{k} + \frac{1}{2k-1}$.

Proof. See Figure 8. We place bundles of $\lfloor k \rfloor$ squares each, as indicated in the figure, starting from the lowest one. This layout uses precisely the $2\lfloor k \rfloor$ x-coordinates $\frac{1}{2} + \frac{i}{2\lfloor k \rfloor - 1}, i = 0, \ldots, 2\lfloor k \rfloor - 1$. This means that every square has x-gap at least $\frac{1}{2\lfloor k \rfloor - 1} \geq \frac{1}{2k-1}$. The y-gap is at least $1/k$ for each square, due to uniform spacing. Both gaps are attained for example by the second-lowest square, so the bound in the lemma cannot be improved for this layout. ◀

Below, we will establish the following result, showing that the simple zigzag layout is asymptotically optimal.

▶ **Theorem 9.** *In the case of uniform spacing, every layout has gap at most* $\frac{1}{k} + \frac{1}{2k-O(\log k)}$.

In proving this, we can restrict to $1/k$-reasonable layouts, the ones achieving gap larger than $1/k$ in the first place. We also assume that $k \geq 2$.

We will start by establishing a crucial fact about such layouts, namely that most of their squares have 3 visible corners. To this end, we are going to upper-bound the number of squares with at least 2 covered corners, eventually enabling us to remove them from the layout while keeping most of the squares.

▶ **Definition 10.** *Given a layout, a* bad square *is one with at least 2 covered corners. A bad square with one vertical side covered is a* standard bad square*; see Figure 9 (left).*

▶ **Lemma 11.** *If a square s has both adjacent squares (in the y-order) in front of it, then s is a standard bad square.*

Proof. If the adjacent squares both have smaller or larger x-coordinate, then they together hide a vertical side of s, see Figure 9 (b). The other case cannot happen in a reasonable layout by Proposition 3; see Figure 9 (c)-(d). Since $k \geq 2$, the adjacent squares actually overlap vertically. ◀

Counting standard bad squares yields a bound for all bad squares.

▶ **Lemma 12.** *For each non-standard bad square, an adjacent square (in the y-order) is a standard bad square.*

Proof. Let s be a non-standard bad square. We distinguish two cases.

The first one is that an upper corner and a lower corner of s are covered. These could be adjacent corners (with some part of the connecting side visible), or antipodal corners as in Figure 10. By Lemma 11, one of the adjacent squares must be behind s; w.l.o.g. it is the next higher one b (blue).

(a)   (b)   (c)   (d)
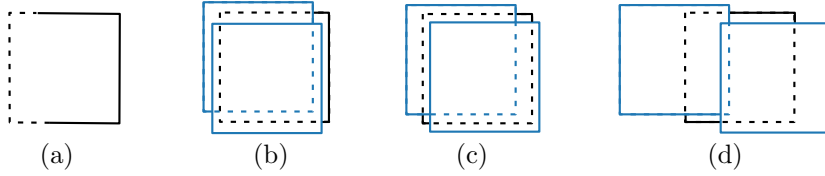
■ **Figure 9** Bad squares: at least two covered corners; A standard bad square ((a) and (b)): one vertical side is covered.
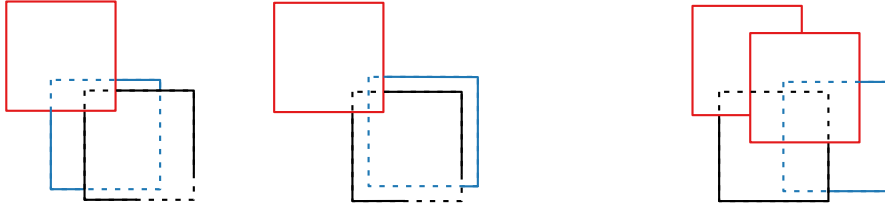
■ **Figure 10** Case 1: A non-standard bad square (black) with an upper and a lower corner covered.

■ **Figure 11** Case 2: A non-standard bad square (black) with two upper or two lower corners covered.

Consider the square r (red) covering the upper corner. Square b is behind s and r, and either "wedged" between them (w.r.t. to both x- and y-coordinate), or "sticking" out. The former case (Figure 10 left) cannot happen, because b would have no gap then, see Proposition 3. In the latter case, b is the required standard bad square (Figure 10 right). This uses that r is higher than b due to uniform spacing.

The second case is that two upper or two lower corners of s are covered, see Figure 11. Let us suppose w.l.o.g. that the two upper corners are covered. Then the upper side of s is covered. This implies that the next higher square b is behind s, as otherwise, s has gap at most $1/k$. Again, b is a standard bad square.   ◄

Through the previous lemma, each standard bad square is "charged" by at most three bad squares (itself and the two adjacent ones).

▶ **Corollary 13.** *For every vertical window $W = [\underline{h}, \overline{h}] \subseteq [\frac{1}{2}, h - \frac{1}{2}]$ of a $1/k$-reasonable layout, the number of bad squares with $y_i \in W$ is at most three times the number of standard bad squares with $y_i \in W' = [\underline{h} - \frac{1}{k}, \overline{h} + \frac{1}{k}]$.*

It remains to count the number of standard bad squares.

▶ **Lemma 14.** *For every vertical window $W = [\underline{h}, \underline{h} + 1]$ of a $1/k$-reasonable layout, there are at most $2(\log k + 1)$ standard bad squares with $y_i \in W$.*

Proof. Let us fix the window. We count the standard bad squares with the left side covered, the overall bound follows by symmetry.

Let $s_1, \ldots, s_\ell$ be these squares; see Figure 12 (left). They must be stacked according to x-coordinate, with squares of lower x-coordinate in front of squares with higher x-coordinate. Indeed, a square s_i in front of a square s_j with smaller x-coordinate would cover a third corner of s_j, and thus a full horizontal side, resulting in no gap (we are using here that the window height is 1). The squares covering the left side of s_i are to the left of s_i and together cover all of s_i in the left half of the strip.

Because the layout is $1/k$-reasonable, each s_i has a part of each of its horizontal sides visible. They are of lengths $\sigma_i \leq \lambda_i \leq 1$ such that $\sigma_i + \lambda_i \geq 1 + 1/k$. Suppose that the squares are ordered by decreasing x-coordinate. We show that the σ_i increase exponentially with i.
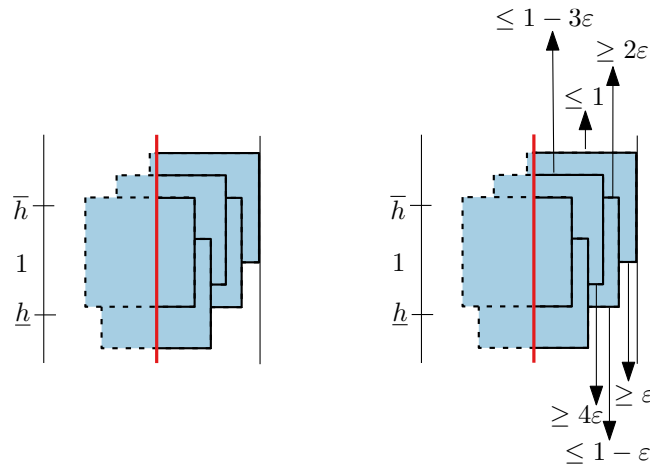
Figure 12 Counting standard bad squares with centers in a vertical window of height 1.
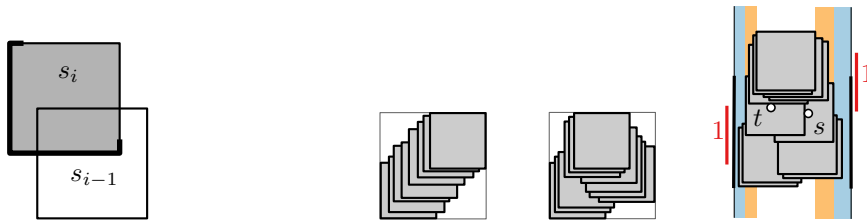
Figure 13 Proof of bitone stacking order.

Figure 14 Two squares of different types in the subwindow.

We have $\lambda_1 \leq 1$, hence $\sigma_1 \geq \varepsilon := 1/k$; see Figure 12 (right). As a consequence, $\lambda_2 \leq 1 - \varepsilon$ (as s_2 is by at least ε further to the left than s_1). Hence, $\sigma_2 \geq 2\varepsilon$. This in turn means that s_3 is by at least $\epsilon + 2\varepsilon$ further to the left than s_1, so $\lambda_3 \leq 1 - 3\varepsilon$ and $\sigma_3 \geq 4\varepsilon$.

Continuing in this fashion, we see that $\sigma_\ell \geq 2^{\ell-1}\varepsilon \leq 1$. This implies that $\ell - 1 \leq \log(1/\varepsilon) = \log k$. ◀

▶ **Corollary 15.** *In a reasonable layout, at most* $6(\log k + 1)$ *squares out of any consecutive* $k - 1$ *squares are bad squares.*

Proof. The centers of $k - 1$ consecutive squares span a horizontal window of height $1 - 2/k$. Using Corollary 13 and the previous lemma, the number of bad squares in this window is at most $3(2(\log k + 1))$. ◀

Hence, by removing $O(\log k)$ squares per bundle of $k - 1$ squares, we obtain a layout with no bad squares left (observe that no surviving square can turn bad by removing squares). Such a layout turns out to have a rather rigid structure.

▶ **Lemma 16.** *After removal of all bad squares from a* $1/k$*-reasonable layout, there is a unique top square (fully visible), and the stacking order is determined: monotone decreasing from the top square towards the highest as well as the lowest square.*

Proof. A square with at least three visible corners (and only such squares remain) is called *down square* if the lower side is fully visible, and *up square* if the upper side is fully visible. A top square is both up and down.

Now let the squares be indexed from lowest to highest. We claim that if s_i is a down square, then s_{i-1} is also a down square that is behind s_i. To see this, consider a down square s_i and the overlapping square s_{i-1} (we have an overlap since we have removed only $O(\log k)$ squares in between); see Figure 13. It is clear that s_{i-1} must be behind s_i, and this in turn implies that s_i is also a down square.

A symmetric statement holds for up squares. Hence, starting from *any* top square, we can go both higher and lower, decreasing stacking height. In particular, we can never encounter another top square. ◀

We now proceed to the proof of Theorem 9, showing that, under uniform spacing, we cannot asymptotically beat the gap of the zigzag layout.

Proof of Theorem 9. We start with the layout obtained after removing all bad squares, as in Lemma 16. W.l.o.g. we assume that the majority of squares is below the top square, and we disregard all squares above. The stacking order then coincides with the y-order.

We now consider a window of height 4. Each square with center in that window is either left or right of the next higher square, and based on this, we call it type L, or type R. We focus on the middle subwindow of height 2. If all squares with centers in this subwindow have the same type, we have a proper staircase of $2k - O(\log k)$ squares; see Figure 14 (left).

Except $O(\log k)$ of them (the ones directly below a removed bad square), all squares have y-gap $1/k$. Let ε be the minimum x-gap among these squares. Then the layout has gap at most $1/k + \varepsilon$. But we know that the sum of all x-gaps is at most 1 (they don't overlap and "live" outside the top square), so the minimum x-gap is $1/(2k - O(\log k))$ which proves the theorem in this case.

The other case is that there are two consecutive squares of different types with centers in the subwindow, see Figure 14 (middle). In this case, we have a generalized staircase. We let s be the lower one and t the higher of the two squares, and we zoom in on the situation; see Figure 14 (right).

Both squares stick out of the ones up to 1 higher than s (otherwise, they can't have 3 visible corners). On the other hand, the squares up to 1 lower than t stick out of both s and t. It follows that – as in the previous case – the x-gaps of all involved squares live outside of the top square among them, and they do not overlap (i.e. they are disjoint). More specifically, the x-gaps of the squares above s live in the orange regions in Figure 14, while the x-gaps of the squares below t live in the blue regions.

In total, we again have $2k - O(\log k)$ squares with a y-gap of $1/k$ within the surrounding window of height 4, so the minimum x-gap is $1/(2k - O(\log k))$. ◀

5   Conclusion

We initiated the algorithmic study of optimizing the visibility of overlapping symbols by finding both a suitable drawing order and a limited displacement. This novel setting leads to various interesting and challenging problems. In this paper we focused solely on unit squares, presented structural insights, as well as several intricate approximation algorithms.

We are curious if the upper bound from Theorem 9 can be improved to one where the $O(\log k)$ term is replaced with a constant. In our approach we derive the bound by eliminating all the *bad* squares from the layout before estimating the gap. Hence, knowing that (in a vertical window of mutually intersecting squares) the number of bad squares can be as large as $\Theta(\log k)$ a radically new approach would be needed to achieve such bound improvement.
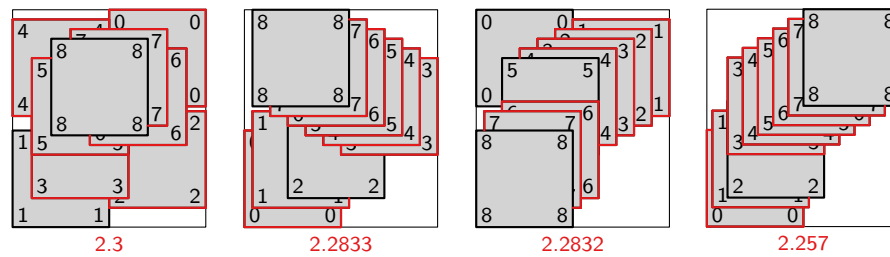
Figure 15 Four layouts for $y = \{0.5, 0.7, 0.8, 1.25, 1.35, 1.45, 1.55, 1.65, 1.75\}$. The minimum visible perimeter is indicated below each layout, the stacking order by the numbers in the corners of each square. From left to right: optimal layout; optimal layout with a stacking order matching the y-order; optimal layout with a stacking order matching the inverse y-order; optimal staircase. These layouts were computed via (I)LPs using a difference of 0.001 to turn strict inequalities into non-strict inequalities.

It is natural to wonder if the stacking order of every optimal solution follows its y-order. However, this is not always the case, refer to Figure 15 for an illustration: the optimal layout (leftmost figure) is better than the best layout when stacking order follows the y-order or the inverse y-order (second and third figures), which is better than the best layout among all staircases (rightmost figure).

An interesting and practically relevant scenario for future work are rectangular symbols. Our algorithms (constructions of layouts) can also be used for this case and yield results of high quality (see Figure 1). However, doing so loses the quality guarantees that we prove for the square case, since the resulting rectangle layouts will not optimize visible perimeter, but a variant of this measure in which horizontal and vertical visible edges have different weights. Even more challenging are settings with differently sized symbols. We leave these question to future work.

References

1 Michael A. Bekos, Benjamin Niedermann, and Martin Nöllenburg. External labeling techniques: A taxonomy and survey. *Computer Graphics Forum*, 38(3):833–860, 2019.
2 Sujoy Bhore, Robert Ganian, Guangping Li, Martin Nöllenburg, and Jules Wulms. Worbel: Aggregating point labels into word clouds. *ACM Transactions on Spatial Algorithms and Systems*, 9(3), 2023. doi:10.1145/3603376.
3 Sergio Cabello, Herman J. Haverkort, Marc J. van Kreveld, and Bettina Speckmann. Algorithmic aspects of proportional symbol maps. *Algorithmica*, 58(3):543–565, 2010.
4 Thomas Depian, Guangping Li, Martin Nöllenburg, and Jules Wulms. Transitions in Dynamic Point Labeling. In *Proceedings of the 12th International Conference on Geographic Information Science (GIScience 2023)*, volume 277 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:19, 2023. doi:10.4230/LIPIcs.GIScience.2023.2.
5 Danny Dorling. *Area Cartograms: their Use and Creation*, volume 59 of *Concepts and Techniques in Modern Geography*. University of East Anglia, 1996.
6 Tim Dwyer, Kim Marriott, and Peter J. Stuckey. Fast node overlap removal. In *Proceedings of the International Symposium on Graph Drawing*, LNCS 3843, pages 153–164, 2005.
7 Jiří Fiala, Jan Kratochvíl, and Andrzej Proskurowski. Systems of distant representatives. *Discrete Applied Mathematics*, 145(2):306–316, 2005.
8 Michael Formann and Frank Wagner. A packing problem with applications to lettering of maps. In *Proceedings of the 7th Annual Symposium on Computational Geometry*, pages 281–288, 1991.

9   Loann Giovannangeli, Frédéric Lalanne, Romain Giot, and Romain Bourqui. Guaranteed visibility in scatterplots with tolerance. *IEEE Transactions on Visualizations and Computer Graphics*, to appear, 2023.

10   Erick Gomez-Nieto, Wallace Casaca, Luis Gustavo Nonato, and Gabriel Taubin. Mixed integer optimization for layout arrangement. In *Proceedings of the Conference on Graphics, Patterns and Images*, pages 115–122, 2013.

11   Daichi Hirono, Hsiang-Yun Wu, Masatoshi Arikawa, and Shigeo Takahashi. Constrained optimization for disoccluding geographic landmarks in 3D urban maps. In *Proceedings of the 2013 IEEE Pacific Visualization Symposium*, pages 17–24, 2013.

12   Kim Marriott, Peter Stuckey, Vincent Tam, and Weiqing He. Removing node overlapping in graph layout using constrained optimization. *Constraints*, 8(2):143–171, 2003.

13   Wouter Meulemans. Efficient optimal overlap removal: Algorithms and experiments. *Computer Graphics Forum*, 38(3):713–723, 2019.

14   Soeren Nickel, Max Sondag, Wouter Meulemans, Stephen Kobourov, Jaakko Peltonen, and Martin Nöllenburg. Multicriteria optimization for dynamic Demers cartograms. *IEEE Transactions on Visualization and Computer Graphics*, 28(6):2376–2387, 2022.

15   Gabriel Nivasch, János Pach, and Gábor Tardos. The visible perimeter of an arrangement of disks. *Computational Geometry*, 47(1):42–51, 2014.

16   Sheung-Hung Poon, Chan-Su Shin, Tycho Strijk, Takeaki Uno, and Alexander Wolff. Labeling points with weights. *Algorithmica*, 38(2):341–362, 2004. `doi:10.1007/s00453-003-1063-0`.

17   Nadine Schwartges, Jan-Henrik Haunert, Alexander Wolff, and Dennis Zwiebler. Point labeling with sliding labels in interactive maps. In Joaquín Huerta, Sven Schade, and Carlos Granell, editors, *Connecting a Digital Europe Through Location and Place*, pages 295–310. Springer International Publishing, 2014. `doi:10.1007/978-3-319-03611-3_17`.

18   Hendrik Strobelt, Marc Spicker, Andreas Stoffel, Daniel Keim, and Oliver Deussen. Rolled-out Wordles: A heuristic method for overlap removal of 2D data representatives. *Computer Graphics Forum*, 31(3pt3):1135–1144, 2012.

19   Mereke van Garderen. *Pictures of the Past – Visualization and visual analysis in archaeological context*. PhD thesis, Universität Konstanz, 2018.

20   Mereke van Garderen, Barbara Pampel, Arlind Nocaj, and Ulrik Brandes. Minimum-displacement overlap removal for geo-referenced data visualization. *Computer Graphics Forum*, 36(3):423–433, 2017.

21   Marc van Kreveld, Tycho Strijk, and Alexander Wolff. Point labeling with sliding labels. *Computational Geometry*, 13(1):21–47, 1999.

22   Claus O. Wilke. *Fundamentals of data visualization: a primer on making informative and compelling figures*. O'Reilly Media, 2019.