Worbel: Aggregating Point Labels into Word Clouds

SUJOY BHORE, Indian Institute of Technology Bombay, India
ROBERT GANIAN, GUANGPING LI, MARTIN NÖLLENBURG, and JULES WULMS,
Algorithms and Complexity Group, TU Wien, Austria

Point feature labeling is a classical problem in cartography and GIS that has been extensively studied for geospatial point data. At the same time, word clouds are a popular visualization tool to show the most important words in text data which has also been extended to visualize geospatial data (Buchin et al. PacificVis 2016). In this article, we study a hybrid visualization, which combines aspects of word clouds and point labeling. In the considered setting, the input data consist of a set of points grouped into categories and our aim is to place multiple disjoint and axis-aligned rectangles, each representing a category, such that they cover points of (mostly) the same category under some natural quality constraints. In our visualization, we then place category names inside the computed rectangles to produce a labeling of the covered points which summarizes the predominant categories globally (in a word-cloud-like fashion) while locally avoiding excessive misrepresentation of points (i.e., retaining the precision of point labeling). We show that computing a minimum set of such rectangles is NP-hard. Hence, we turn our attention to developing a heuristic with (optional) exact components using SAT models to compute our visualizations. We evaluate our algorithms quantitatively, measuring running time and quality of the produced solutions, on several synthetic and real-world data sets. Our experiments show that the fully heuristic approach produces solutions of comparable quality to heuristics combined with exact SAT models, while running much faster.

CCS Concepts: • **Human-centered computing** → **Geographic visualization**; Empirical studies in visualization; • **Theory of computation** → *Computational geometry;*

Additional Key Words and Phrases: Labeling, word clouds, categorical point data

ACM Reference format:
Sujoy Bhore, Robert Ganian, Guangping Li, Martin Nöllenburg, and Jules Wulms. 2023. Worbel: Aggregating Point Labels into Word Clouds. *ACM Trans. Spatial Algorithms Syst.* 9, 3, Article 19 (August 2023), 32 pages.
https://doi.org/10.1145/3603376

Source code, benchmark data generator and real-world data extractor at https://dyna-mis.github.io/geoWordle/. An extended abstract of the article appeared in the proceedings of the 29th International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL'21).

We acknowledge support by the Austrian Science Fund (FWF) under grants Y1329 (Robert Ganian, Jules Wulms) and grant P 31119 (Guangping Li, Martin Nöllenburg, and Jules Wulms), and by the Vienna Science and Technology Fund (WWTF) [10.47379/ICT19035] (Martin Nöllenburg, Jules Wulms).

Authors' addresses: S. Bhore, Main Gate Rd, IIT Area, Powai, Mumbai, Maharashtra 400076; email: sujoy.bhore@gmail.com; R. Ganian, G. Li, M. Nöllenburg, and J. Wulms, Technische Universität Wien, Institute of Logic and Computation, Favoritenstraße 9–11, E192-01, 1040 Wien, Austria; emails: rganian@gmail.com, {guangping, noellenburg, jwulms}@ac.tuwien.ac.at.

2374-0353/2023/08-ART19 $15.00
https://doi.org/10.1145/3603376

1 INTRODUCTION

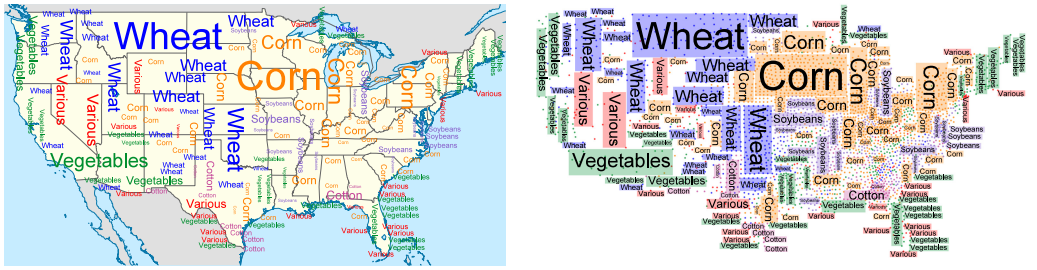
Labeling graphical features in maps is a classical problem in cartography and **geographic information systems** (**GIS**), and has been extensively studied [35, 38]. The map features to be labeled range from points and lines to areas. Point labeling was introduced in computational cartography about 30 years ago [20, 26] and has since been studied in many different settings, e.g., [3, 22, 30, 32]. In many real-life applications, however, which deal with massive data, there can be many repetitions of labels. For instance, in Twitter data, where hashtags are used to indicate the topic of a tweet, or in categorical point-of-interest data, a few common topics appear many times. Geographic visualizations of huge social network data, especially for anomaly detection and visual analytics [25, 33], normally require information aggregation.

A *tag map* is a visualization technique that achieves this by placing a categorical label for clusters of points in a small area at a fixed pre-determined position [31]. These clusters are shown using an enlarged label of the predominant category in the area. While this may lead to many overlapping labels, tag maps have been extended to prevent these overlaps, and adapt the label sizes in more sophisticated ways [28, 33]. However, in tag maps, a point can be unlabeled, far away from its representative label, or even covered by a label of a different category. A disadvantage of tag maps is that they provide no guarantees on the number of such misrepresented points or their distances to the nearest correct label. In this sense, the precision guarantee of point labeling is lost in tag maps.

Another popular method used to aggregate text-based data in information visualization is word clouds, where typically the objective is to highlight the most frequent keywords in a text [37]. More frequent keywords are visualized using a larger font size. This technique has been extended to work with geo-spatial point data that have keywords associated with each point [11]. These *geo word clouds* place keywords in areas that contain many points with the associated keyword. For each keyword, the associated points are clustered, and for each cluster a good label placement is computed in terms of rotation and absolute position. The labels are scaled based on the number of points in the associated cluster, and are placed in decreasing order of their scaling factor. Whenever the currently placed label overlaps an earlier placed label, the current label is shrunk a little bit, and a new position is determined. The label is placed again, after existing labels with larger scaling factors are considered first. The shrinking ensures that it becomes easier to place the label, albeit further from the intended position. While such word clouds allow keywords to be displayed in a non-overlapping manner with a large degree of freedom, for example in terms of orientations and absolute placements, they have similar drawbacks as the aforementioned tag maps: there are no guarantees on the precision of such labeling, as the best placement for a label can be blocked by an earlier placed label (see, e.g., Figure 7 in [11]).

Therefore, we are facing the challenge to find a hybrid, label-based visualization that simultaneously retains the precision of classical point labeling, where labels are placed exactly at their corresponding points, and produces aggregated word-cloud-like labels that represent spatial patterns of keyword occurrences well.

Contributions. In this work, we study a hybrid visualization that combines aspects of word clouds and point labeling, which we call a *Worbel*, see Figure 1 for an example. The input consists of a set of points in the plane, each of which has a category associated with it. We want to place disjoint textual labels, representing the categories, such that the labels spatially represent the underlying data points of the same category. We develop a formalization of this task as the problem of finding a small set of disjoint axis-aligned category rectangles which cover points of each category while adhering to constraints that bound the amount of *misrepresentation*, the *aspect ratio*, and require a *minimum size*; we call this the **Rectilinear Point Feature Aggregation**



(a) (b)

Fig. 1. Worbel visualizations of the dominant crop harvested in each county of the United States. (a) A Worbel overlay on a map of the US. (b) The same Worbel drawn over the data points, with rectangles outlining computed area per label.

(**RPFA**) problem. As our first theoretical contribution, we develop a non-trivial reduction showing that RPFA is NP-hard and remains so even under severe restrictions.

Since this rules out an efficient exact algorithm for the problem, we turn our attention toward robust heuristics. We first compute a set of candidate rectangles, which we use to cover the points. We then present a MaxSAT-model for finding an optimal set of disjoint rectangles among our candidates, as well as a simple but highly effective heuristic algorithm designed to find a sufficiently good (but not necessarily optimal) selection of disjoint rectangles in our candidate set. In a quantitative evaluation we show that this heuristic approach computes solutions that are close in quality to the MaxSAT-model, but requires a much lower computation time. Our evaluation includes a case study on real-world data, in which we demonstrate the visual qualities of the Worbel technique.

Related Work. Map labeling articles [3, 10, 36] often focus on the computation of the maximum independent set of geometric objects as an underlying combinatorial optimization problem. While this turned out to be an appropriate abstraction for point-feature labeling, it may omit many labels and thus only show a fraction of the input labels due to geometric packing constraints. In our setting, even though we are interested in a labeling that represents the underlying point features well, we provide an aggregated labeling of categorical data with many repeated labels. Specifically, we label multiple points of the same category with a single label and therefore produce a labeling that is closer to an area-labeling [7] of implicitly defined areas of homogeneous points.

Word clouds have been used as a key tool for text-based visualization, where the goal is to highlight prominent keywords in large amounts of text, emphasizing more important or frequent words by a larger font size [8, 15, 37]. Moreover, the study of word clouds and tag maps has been extended to more specialized settings, such as metro wordles [24] and time-varying tag maps [29].

Besides text-based geographic visualizations, there are other types of thematic maps [19] to portray spatial patterns of categorical point and area data, e.g., choropleth and chorochromatic maps, dot maps, and generally maps placing symbols and marks that convey data attributes by visual variables such as size, texture, color, or shape, and whose meaning is provided via lookups in a map legend.

The RPFA problem has its roots in geometric covering problems, and many variants of geometric covering problems have been investigated over the years, e.g., (p, k)-box covering [4], class cover problems [9], and red-blue set cover [6, 12, 14]. Moreover, geometric covering problems have been widely studied in various algorithmic paradigms such as approximation algorithms (see [2, 16, 18]) and parameterized algorithms (see [5, 6]). Most of these works, however, do not consider the case where the covering shapes must be disjoint. This makes our problem particularly unique, and none of these algorithms can be directly applied in our context.

Article Organization. We introduce the RPFA problem formally in Section 2 and show that it is NP-hard in Section 3. We then present the MaxSAT-model and our greedy heuristic in Section 4. In Section 5 the results of our quantitative evaluation are presented. Finally, we discuss the presented results and potential improvements in Section 6.

2 THEORETICAL MODEL

We begin by providing a formalization of the problem that underlies the task of aggregating point features in a Worbel.

Problem 1. *In Rectilinear Point Feature Aggregation (RPFA), we are given a set P of points in the plane $\mathbb{R}^2$, a set L of different labels, and a label function $\ell\colon P \to L$ which assigns each point in P to a label in L. The task is to cover a maximum-cardinality subset of P with a minimum-cardinality set $\mathcal{S}$ of pairwise-disjoint axis-aligned rectangles, while adhering to the parameters ρ_l, ρ_u, t, and f described below. Additionally, each rectangle $R \in \mathcal{S}$ is assigned a label $l_R \in L$ and each point may only be located in a rectangle of the same label.*

In a Worbel, label l_R will be printed inside each rectangle R, scaled to the size of R and rotated to match the major axis. Each rectangle $R \in \mathcal{S}$ must additionally adhere to the following constraints; see Figure 2.

Aspect ratio. The aspect ratio a_R of R should be close to the aspect ratio a_{l_R} of the bounding box of label l_R. We introduce parameters ρ_l, ρ_u, with $0 \leq \rho_l < 1 < \rho_u$ as the lower and upper bounds on the ratio between a_R and a_{l_R}. Specifically, we ensure that $\rho_l \leq \frac{a_R}{a_{l_R}} \leq \rho_u$ for each rectangle $R \in \mathcal{S}$.

Tolerance for misrepresentation. The points covered by R should "predominantly" be assigned to the label l_R of R. To this end, we introduce a tolerance threshold t, which quantifies how many points in R can be of a different label. The number of points with a label different from l_R is at most $\min(t, 0.5 \cdot |R|)$, when R covers $|R|$ points.

Minimum font size. We enforce a minimum font size f to ensure that labels remain readable on screen, and cannot be scaled down arbitrarily. To this end, the minor axis of R is not allowed to be smaller than f.

Observe that it may not be possible to cover all points by disjoint rectangles due to the constraint on the minimum font size; in these cases, we simply aim at covering as many points as possible. we say that a set $\mathcal{S}$ of disjoint rectangles is *viable* if it covers a maximum-cardinality subset of P while satisfying all the constraints specified above. Hence, in its full generality, an instance of RPFA is a tuple $(P, L, \ell, \rho_l, \rho_u, t, f)$, and an *optimal solution* is a minimum-cardinality viable set $\mathcal{S}$ of rectangles.

3 PROBLEM COMPLEXITY

For complexity-theoretic purposes, let RPFA* be the decision version of RPFA; there, we are additionally given an integer $k \in \mathbb{N}$ and ask whether there exists a set $\mathcal{S}$ of viable rectangles of cardinality at most k. It is easy to observe that RPFA* is in NP, since we can non-deterministically guess a set of at most k rectangles and check whether these rectangles are pairwise disjoint and cover all the points in polynomial time. As one of our main theoretical contributions, we will show that RPFA* is not only NP-hard, but it remains NP-hard even when restricted to a very specific subset of inputs. In particular, let RPFA_0 be the restriction of RPFA* to inputs where $L = \{\text{red}, \text{blue}\}$, $\rho_l = 0$, $\rho_u = \infty$, and $t = f = 0$. Intuitively, RPFA_0 is the restriction of RPFA* where the aspect ratio and minimum font size constraints are disregarded, no points are misrepresented, and we have only two labels.

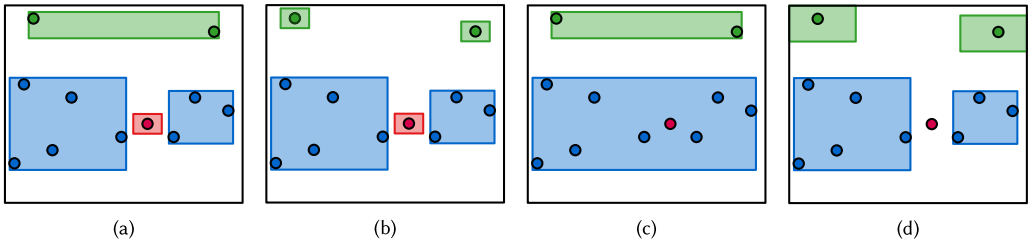


Fig. 2. Solutions to RPFA with varying parameters. Distinct labels are indicated as different colors. (a) Parameters $\rho_l = 0$, $\rho_u = \infty$, and $t = f = 0$. (b) Lowered ρ_u w.r.t. (a). (c) Increased t to 1 w.r.t. (a). (d) Increased f w.r.t. (a).

Theorem 3.1. *RPFA$_0$ is* NP*-complete.*

To prove Theorem 3.1, we reduce from the vertex cover problem on 3-regular planar graphs. The vertex cover problem takes a graph $G = (V, E)$ as input and asks to find a minimum subset $S \subseteq V$ such that for each edge $e \in E$, at least one of the vertices at the endpoints of e are chosen in S, i.e., S *covers* E. This problem is already NP-hard for 3-regular planar graphs [21], which are graphs that have only vertices with three incident edges, and that can be embedded in the plane without edge crossings. However, we do not give a direct reduction but rather reduce via an intermediate problem, called **Disjoint Box Covering in a Rectilinear Polygon** (**DBCR**).

Problem 2. *In DBCR, we are given a rectilinear polygon ψ, which has only horizontal/vertical sides and possibly has holes, a set Γ of n points inside ψ (which we call* elements *for disambiguation), and a bound ω. The question is whether there exists a set $\mathcal{Q}$ of at most ω pairwise disjoint axis-aligned rectangles such that each element in Γ is contained in (i.e.,* covered by*) a rectangle in $\mathcal{Q}$ and each rectangle is fully contained inside ψ.*

Two reflex corners x, y of ψ are called *opposite* if the unique axis-aligned rectangle U which has x and y as its corner points, is fully contained in ψ, and only intersects the boundary of ψ in x and y; in this case, we call U the *x-y-spanning rectangle* (see Figure 5 for an example of such a rectangle).

Theorem 3.2. *DBCR is* NP*-hard, even when restricted to instances with the following properties:*

- *each x-y-spanning rectangle contains at least one element,*
- *each corner of ψ and element lie on a finite integral* embedding grid *whose size is polynomial in the input size, and*
- *each pair of two elements, each pair of one element and one corner of ψ, and each pair of corners not touching the same boundary segment have distinct x- and y- coordinates (i.e., they lie in general position).*

3.1 Proof of Theorem 3.2

The remainder of this section is devoted to show the NP-hardness of the DBCR problem. We do a reduction similar to the one by Chan and Hu for red-blue set cover [14], using the following lemmata.

Lemma 3.3 ([34]). *Every planar graph $G = (V, E)$ of maximum degree at most* 4 *has an orthogonal grid drawing on an $O(|V|) \times O(|V|)$ grid.*

Lemma 3.4 (Folklore). *Given a graph $G = (V, E)$ and an edge $e \in E$, define a new graph G' obtained from G by subdividing e through the addition of two new vertices. Then the size of a minimum vertex cover of G' is exactly the size of a minimum vertex cover of G plus* 1.

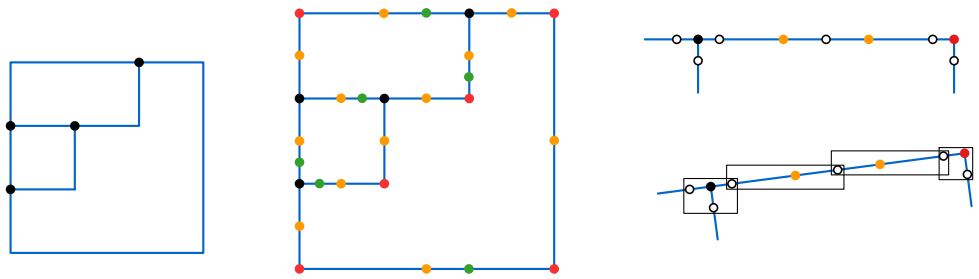


Fig. 3. The reduction on graph $G = K_4$. (a) An orthogonal grid drawing δ_G of G. (b) The dummy vertices are added in δ_G to the bends (red) and on the edges, both to prevent multiple corner vertices incident to an edge (yellow), and for parity (green). (c) The cover rectangles, for black, red, yellow and green vertices, cover all (white) points in Γ and are drawn on the shifted grid.

We reduce from the vertex cover problem on 3-regular planar graphs, which is known to be NP-hard by Garey and Johnson [21]. Given a 3-regular planar graph G with n vertices, we create an orthogonal grid drawing δ_G by Lemma 3.3; see Figure 3(a). The proof for Lemma 3.3 is constructive, and the procedure takes polynomial time: The vertices are placed consecutively, and each vertex is placed in a predetermined position with respect to the already placed vertices. We can use any textbook planarity testing algorithm, specifically a vertex addition algorithm [27], to find an appropriate placement order for the vertices. We now construct a new graph G' by adding several dummy vertices in δ_G.

Firstly, we add a dummy vertex on each bend of δ_G, to create a straight-line drawing. We call a vertex v in δ_G a *corner* vertex, if there exists a pair of perpendicular line segments in δ_G that meet at v. Then, for each edge e connecting two corner vertices, we add a dummy vertex in the middle of e. This results in the new graph G', where each edge is incident to at most one corner vertex. To apply Lemma 3.4, for each original edge e in G, we check whether there is an even number of dummy vertices on e, including dummy corner vertices. If not, then we add a dummy vertex anywhere on the edge, to ensure the vertex cover of G' changes predictably (see Figure 3(b)). Let $\delta_{G'}$ be the resulting straight-line drawing of G'. We then place $\delta_{G'}$ on an $O(|V|) \times O(|V|)$ regular grid in which each cell is a unit square such that each vertex of G' is placed on a grid point.

To construct the element set Γ in the DBCR instance from $\delta_{G'}$, we replace each edge in $\delta_{G'}$ by an element. If an edge e is incident to a corner vertex v, we put an element at $\frac{1}{4}$ of grid length from v on e. Otherwise, we create an element in the middle of the edge. Now we shift the elements such that there are no two elements on the same horizontal or vertical line. To achieve this, we map the drawing $\delta_{G'}$ to an orthogonal grid counterclockwise rotated by an angle γ. This angle depends on the length l of the longest edge of $\delta_{G'}$: we choose $\gamma < \arcsin(\frac{1}{4l})$.

For each vertex v in G', we draw a rectangle B_v, such that B_v is the minimum bounding box covering v and the elements on edges of v (see Figure 3(c)). By our choice of γ these rectangles overlap only at elements of Γ (see Lemma 3.5).

We say that B_v is the *cover rectangle* of the vertex v. For each corner vertex, we add a margin of length $\frac{1}{2}\epsilon$ to its cover rectangle, where ϵ is an infinitesimally small distance. This ensures that the covered elements are not on its boundaries and the rectangles grow by ϵ both horizontally and vertically. Intuitively, there are two types of cover rectangles, small rectangles for corner vertices and long rectangular bars otherwise. We can then make the following observations.

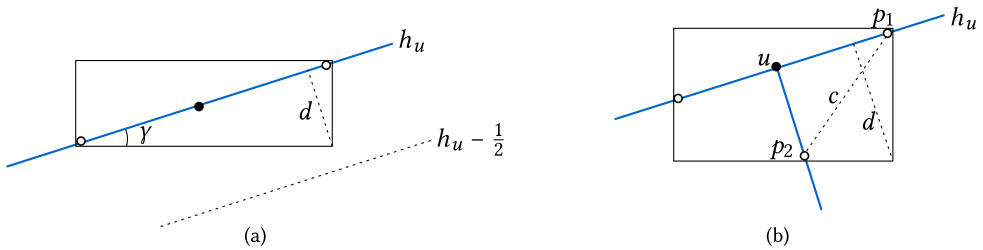


Fig. 4. Illustration for the proof of Lemma 3.5.

Lemma 3.5. *Two cover rectangles intersect if and only if their corresponding vertices are adjacent. For each edge of G', only the rectangles of its two incident vertices intersect with it and the intersection contains the element on the edge.*

Proof. By the construction, cover rectangles of adjacent vertices intersect and the intersection contains the element of the corresponding edge. Consider two non-adjacent vertices u, v in G', it is sufficient to prove that their cover rectangles B_u, B_v do not overlap. If u, v lie on the same grid line, the claim is obvious. Suppose u, v lie on different grid lines h_u and h_v, respectively. We assume, without loss of generality, that these lines are horizonal and h_u is above h_v.

Now we show that B_u is above the line $h_u - \frac{1}{2}$. If u is not a corner vertex, every element in the cover rectangle B_u is placed on an incident edge of u; see Figure 4(a). Since the distances from the elements covered by B_u to the vertex u are bounded by l, where l is the length of the longest edge of $\delta_{G'}$, the diagonal of B_u is bounded by $2l$. By the choice of $\gamma < \arcsin(\frac{1}{4l})$, the minor axis of B_u is bounded by $\frac{1}{2}$. The distance d between a corner of B_u and h_u is also bounded by $\frac{1}{2}$, since the minor axis of B_u is the hypotenuse of a right-angled triangle with side length d. Thus, B_u is above $h_u - \frac{1}{2}$.

If u is a corner vertex, we know that each element covered by B_u is placed on an edge incident to the cover vertex u with distance of $\frac{1}{4}$ of grid length to u; see Figure 4(b). Here, the elements p_1, p_2 covered by B_u are at a distance of $\frac{1}{4}$ of grid length from u, thus the distance c between p_1 and p_2 is strictly bounded by $1/2$. Again using right-angled triangles, we find that the distance d from a corner of B_u to h_u is bounded by the minor axis of B_u, which is further bounded by c. Thus, B_u is above the line $h_u - \frac{1}{2}$. By the same argument, we get that B_v is totally below the line $h_v + \frac{1}{2}$.

Since in no case B_u and B_v can reach halfway into the incident grid cells, B_u and B_v do not overlap. □

The union of cover rectangles builds the rectilinear polygon ψ in our DBCR instance. Note that the constructed DBCR instance has the listed properties in Theorem 3.2.

Lemma 3.6. *Given an arbitrary rectangle R in ψ that covers a subset Γ' of element set Γ, the minimum bounding box of Γ' is inside a cover rectangle.*

Proof. To prove this, it is sufficient to show that for two elements a, b which cannot be covered by a cover rectangle, their minimum bounding box is not inside ψ. Suppose that a and b are neither on the same horizontal nor vertical grid line. Observe that cover rectangles of corner vertices are the only cover rectangles intersecting multiple grid lines. Since a and b are not covered by a single cover rectangle (of a corner vertex), there are two cover rectangles that are stabbed by different grid lines and cover a and b, respectively. Therefore, the minimum bounding box of a and b has intersections with the boundaries of ψ defined by those cover rectangles. If a and b are on the same (horizontal or vertical) grid line, we assume w.l.o.g. that a and b are on the same horizontal grid line h. Observe that the top edges of cover rectangles stabbed on h are arranged in a monotonically

increasing or decreasing y-order (see the "staircase" in Figure 3(c)). Furthermore, the elements on h are placed in a similar monotonic y-order. Therefore, for two elements a and b that are on h and not shared by a cover rectangle, their minimum bounding box must intersect the boundary of ψ, in particular the top edge of the lower cover rectangle, and hence this bounding box is not inside ψ. In both cases, the boundary of ψ is intersected by the minimum bounding box of a and b, and thus the claim holds. □

Thus, after shifting, the elements (and the corresponding cover rectangles) on a horizontal/vertical grid line are in a monotonic order. For two elements that are not covered by the same rectangle B_v of any vertex v, there is no rectangle inside ψ that covers these elements at the same time.

The correctness of the reduction is now straightforward. A vertex cover in G' corresponds to a rectangle set $\mathcal{S}$ of the same size, each of which is inside the polygon ψ and covers all the elements of Γ. By Lemma 3.5, if two cover rectangles intersect each other, their corresponding vertices are adjacent. Due to our construction, no two corner vertices are adjacent. Therefore, each intersection involves at least one non-corner vertex. Consider a rectangle $R \in \mathcal{S}$ whose corresponding vertex is not a corner vertex. If the rectangle R intersects other rectangles in $\mathcal{S}$, we could eliminate the intersection by shrinking long rectangle R, while still covering all the elements. After eliminating all the intersections in $\mathcal{S}$, $\mathcal{S}$ is a set of pairwise disjoint rectangles that cover the element set Γ. In the other direction, we are given a set $\mathcal{S}$ of disjoint rectangles inside ψ that cover all the elements of Γ. By Lemma 3.6, each rectangle in $\mathcal{S}$ is completely inside a cover rectangle in the polygon ψ. We select all the cover rectangles that contains rectangles in $\mathcal{S}$. These rectangles cover all the elements and their corresponding vertices in G' cover all the edges in G', and thus build a vertex cover in G'.

3.2 Proof of Theorem 3.1

We provide a polynomial reduction from DBCR to RPFA$_0$. Intuitively, the main difference between the two problems is that while in DBCR there is a clearly defined bounding polygon ψ, in RPFA$_0$ the placement of rectangles is restricted exclusively by the presence of points with a different label (which themselves need to be covered by a rectangle of the other color). To overcome this difficulty, the reduction will use a construction that simulates the presence of ψ; this is also where the property of spanning rectangles mentioned earlier will come into play. The reduction will take as input an instance $\mathcal{I}' = (\psi, \Gamma, \omega)$ of DBCR with the three properties of Theorem 3.2 and construct an instance $\mathcal{I}$ of RPFA$_0$. Figure 5 illustrates one such instance $\mathcal{I}$. We begin by refining the resolution of the embedding grid by a factor of 10; the sole purpose of this is to ensure that the coordinates of each element and corner differ by at least 10. We will call the points of the embedding grid that lie on the boundary of ψ *boundary points*. We iteratively partition all the points of the instance $\mathcal{I}$ outside of ψ, but inside the embedding grid, into layers as follows. The first layer contains all points which are adjacent to a boundary point in the 8-point neighborhood topology (i.e., these are either axis-adjacent or diagonally adjacent to a boundary point). Once we have identified the set of layer-i points, we define the set of layer-$(i + 1)$ points by repeating this procedure—in particular, a point is in layer $i + 1$ if it does not belong to any layer up to i and is adjacent to a point in layer i in the 8-point neighborhood topology.

We are now ready to construct the instance $\mathcal{I}$ of RPFA$_0$, which will be the output of our reduction. Initially, each element in Γ will be added to P as a red point. We then add into P all points in odd layers as red points, and all points in even layers as blue points. Observe that this creates a sequence of alternating red and blue axis-aligned polygonal contours around ψ, see Figure 5. Let δ be the minimum number of straight-line segments required to cover all edges of

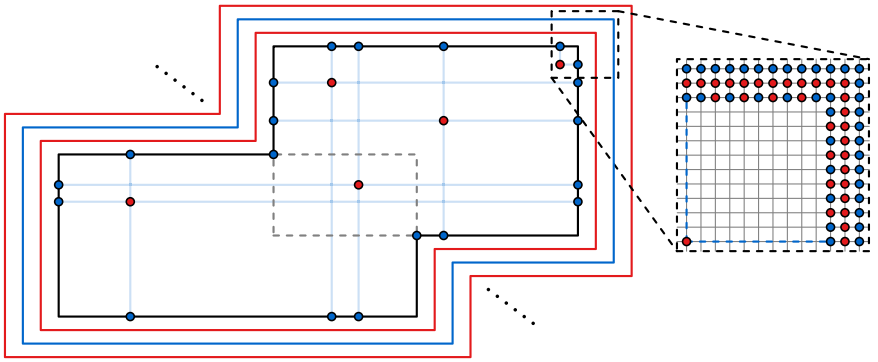


Fig. 5. Construction of $\mathcal{I}$: the initial grid is refined by a factor 10, and there are alternating layers of red and blue points around initial polygon ψ. Additionally, the boundary consists of (mostly) alternating red and blue points, with irregularities around corners and grid lines on which elements lie. Note that each x-y-spanning rectangle will contain a red point.

these contours, and notice that δ can be computed in polynomial time via a straightforward greedy procedure.

Intuitively, we now have an instance of RPFA_0 where we can ascertain precisely how many rectangles are needed to cover the points in the layers as long as the points in layer 1 are not allowed to be covered by rectangles that intersect ψ. Moreover, under the same assumption about the set of rectangles covering the points in layer 1, the internal area of ψ could be used exclusively for rectangles covering Γ, precisely matching the behavior of the original instance $\mathcal{I}'$. Unfortunately, there is currently nothing that would prevent the rectangles covering the points in layer 1 from entering ψ; that is where the next, crucial step of the reduction comes in.

We say that a boundary point is *important* if it is either a corner of ψ, or if it has the same x- or y-coordinate as a point in Γ. All important boundary points will be added to P as blue points (see Figure 5). All remaining boundary points will also be added to P in a way, which ensures that every two such points that lie "opposite" to each other on the boundary of ψ receive a different color. In particular, each non-important boundary point that lies on:

— a bottom boundary of ψ is red $\iff$ its x-coordinate is odd;
— a top boundary of ψ is blue $\iff$ its x-coordinate is odd;
— a left boundary of ψ is red $\iff$ its y-coordinate is odd;
— a right boundary of ψ is blue $\iff$ its y-coordinate is odd.

The result is that the boundary points form a checkered red-blue pattern, but with some local irregularities around the important points (see Figure 5). Finally, we set k to be equal to the sum of (a) the number δ of line segments for contours outside of ψ (b) the bound ω on rectangles to be used in the original instance $\mathcal{I}'$, and (c) the total number σ of color switches that occur on the boundary points when performing a walk along each boundary segment of ψ.

Running time and correctness. Clearly, the construction described above can be carried out in polynomial time, and hence all that remains is to argue correctness. For the forward direction, assume that $\mathcal{I}'$ was a *yes-instance* of DBCR and admits a solution $\mathcal{Q}$ (i.e., a set of rectangles covering Γ, contained in ψ, and of size at most ω). We can then construct a solution $\mathcal{S}$ for $\mathcal{I}$ by

(1) covering all the points in the layers using δ-many rectangles.
(2) having one rectangle copy the placement of each of the at most ω-many rectangles in $\mathcal{Q}$.

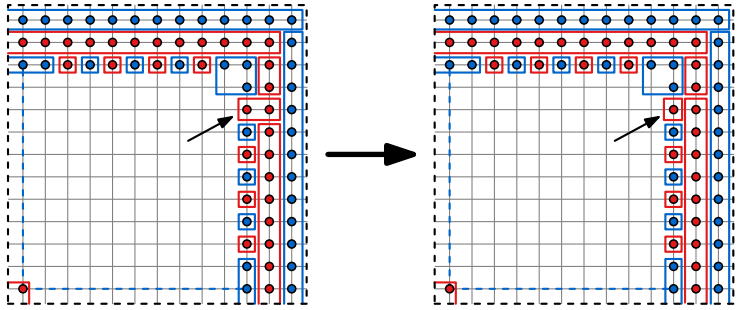


Fig. 6. A local view of a solution $\mathcal{S}$ for DBCR, which is changed to a solution of the same cardinality, but having one less rectangle that covers both boundary points and points in layer 1. The small arrows point to the local change in solution.

(3) using one rectangle to cover each consecutive set of red or blue boundary points that occur on a walk around each boundary segment of ψ (σ-many in total)

It is easy to verify that this construction produces a set $\mathcal{S}$ of at most k rectangles which cover all the points in P.

For the backward direction, assume that $\mathcal{I}$ admits a solution $\mathcal{S}$. First of all, we will argue that $\mathcal{S}$ can be assumed to use δ-many rectangles to cover all points that lie in the individual layers without interfering with ψ. For each layer other than the first, each point can only lie in a rectangle with other points in the same layer which makes the claim obvious, but a careful argument is needed for the first layer (see Figure 6).

To this end, assume that $\mathcal{S}$ covers some red point p in layer 1 with a rectangle T that also contains a red point q that does not lie in layer 1; this point would have to be a boundary point. But by the coloring used for the boundary points, T can then only contain at most one red boundary point, which in this case is q, and no red points in layer 1 other than p. Let a and b be the two unique red points on layer 1 that are adjacent to T when walking along layer 1, and notice that the grid refinement we performed at the very beginning prevents a and b from both being corners simultaneously; without loss of generality, let us assume that b is not a corner. Then the two axis-neighboring points of b outside of layer 1 must be blue. Let B be the rectangle in $\mathcal{S}$ which contains b, and notice that B can only contain points in layer 1. We can now alter $\mathcal{S}$ as follows: extend B to cover all red points in T, and restrict T only to those points which lie outside of layer 1 (see Figure 6).

We have now obtained a new solution which has an equal size as the original $\mathcal{S}$, but contains one less occurrence of a rectangle containing points in layer 1 and also on the boundary. By iterating this procedure, we arrive at an equivalent solution $\mathcal{S}$ such that no rectangle containing points in layers intersects ψ; these rectangles hence form walks along the individual layers and $\mathcal{S}$ contains at least δ-many of them.

Now, consider the rectangles in $\mathcal{S}$ which contain at least one boundary point. Crucially, by combining

— the construction of the colors on the boundaries,
— the fact that each spanning rectangle in $\mathcal{I}'$ contains at least one point and boundary corner points are blue, and
— the fact that each pair of corners of ψ that do not share a segment on the boundary of ψ lie in general position (i.e., have distinct x- and y- coordinates),

we obtain that every rectangle in $\mathcal{S}$, containing at least one boundary point, can only contain other boundary points, and that these must lie in a consecutive strip of points along the boundary;

in other words, without loss of generality, we may assume that such a rectangle is either a line segment which follows the boundary, or a 2×2 square containing one blue corner point along with two adjacent blue points[1] (see Figure 6).

As a consequence, we obtain that all rectangles in $\mathcal{S}$ can be partitioned into three pairwise disjoint sets: those used to cover the points in the layers, those used to cover the boundary points, and those used to cover the points inside ψ. Since $|\mathcal{S}| \leq k$ and the first two subsets of $\mathcal{S}$ have a cardinality of at least δ and σ, we obtain that the number of rectangles in the third set is at most ω. But then the set of rectangles used by $\mathcal{S}$ to cover the points inside ψ is a solution for $\mathcal{I}'$, as required. This concludes the proof showing that RPFA_0 is NP-hard.

In view of Theorem 3.1, we cannot hope for an exact algorithm for RPFA with a polynomial-time runtime guarantee. We, therefore, turn to efficient heuristics, and show that these can produce good Worbels in many scenarios of interest.

4 ALGORITHMS FOR RPFA

We propose to solve RPFA heuristically: We start by computing a set of candidate rectangles under the parameter constraints of RPFA. Given this candidate set, we introduce a greedy heuristic and an exact SAT model for computing a viable set of disjoint rectangles among the candidates. We first discuss how to find an appropriate set of candidate rectangles, given the aspect ratio bounds ρ_l, ρ_u, tolerance t, and minimum font size f.

4.1 Computing Candidate Rectangles

A naive idea for computing the candidate set $\mathcal{R}$ of rectangles would be to simply consider the set of all rectangles whose left, right, bottom and top boundary touches some point in P. This would result in $\Theta(|P|^4)$ combinatorially different rectangles, from which a subset $\mathcal{S}$ of at most $|P|$ disjoint rectangles to cover P will be selected. This set of rectangles always contains an optimal solution to RPFA, if aspect ratio and font size are unrestricted: $\rho_l = 0$, $\rho_u = \infty$, and $f = 0$.

Since we want the candidate rectangles to adhere to any setting of the input parameters, the naive approach does not always lead to an appropriate set of candidate rectangles: On the one hand, we often need to consider far less rectangles, as many rectangles computed by the naive approach may violate the input parameters. On the other hand, the sides of the rectangles in an optimal solution do not necessarily touch some point in P, for example when a minimum font size is enforced. This results in a potentially infinite set of candidate rectangles, because of variable side lengths and translations of rectangles. Hence, we instead consider a subset of the rectangles computed by the naive approach, which certainly adheres to the input parameters, and we additionally try to account for some of the variability in side lengths and translations. This new set of rectangles does not necessarily contain all rectangles of an optimal solution, but we show in Section 5 that this approach suffices to find Worbel visualizations of good quality.

For each pair of distinct points $p_i, p_j \in P$ we compute a set $\mathcal{R}_{i,j}$ of candidate rectangles as follows. We first compute the smallest bounding box R_{base} of p_i and p_j. This rectangle will be the base for a set of rectangles that extends leftward and rightward, inside the horizontal strip defined by p_i and p_j (see Figure 7).

We first check which label $l \in L$ is the predominant label among the points in R_{base}, and we additionally make sure that the number of points with a label different from l does not exceed

[1] A careful reader may notice that this degenerate case may only occur for points in the top-right corner, but such rectangles do not intersect the space required for covering the red points in ψ due to the fact that each red point has a gap of at least 10 grid points from every corner point due to the grid refinement used.

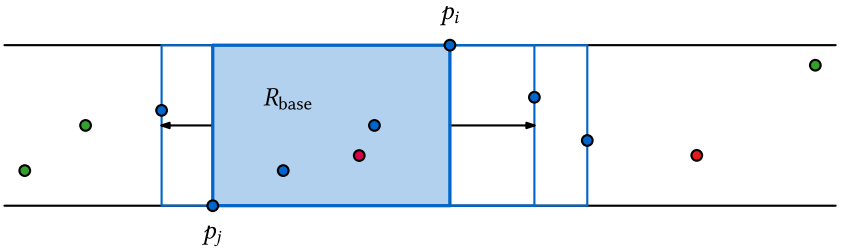


Fig. 7. Horizontal strip defined by p_i and p_j, in which R_{base} will be extended leftward and rightward, until one of the input constraints (here misrepresentation) is violated.

$\min(t, 0.5 \cdot |R|)$. If R_{base} fails the tolerance check, we move to the next pair $p_i, p_j \in P$, otherwise we assign $l_{R_{\text{base}}} := l$ and proceed.

Rectangle R_{base} is now extended leftwards, creating a new candidate for each additional point covered by an extended rectangle. Observe that this operation has the following consequences for the aspect ratio, tolerance, and minimum font size. The aspect ratio of vertically oriented rectangles will increase, until they become squares and the orientation becomes horizontal. From that point on, the aspect ratio will decrease, and we stop extending when the aspect ratio of an extended rectangle R, with label l_R and label aspect ratio a_{l_R} falls below $\rho_l \cdot a_{l_R}$. The number m of points with a label different from l_R may both increase or decrease, since newly covered points may be of label l_R or of a different label, respectively. However, we stop extending as soon as m exceeds $\min(t, 0.5 \cdot |R|)$. The font size will monotonically increase: for vertically oriented rectangles only the height increases and after flipping to the horizontal orientation only the width increases. Therefore font size does not influence when we stop extending. Thus for all extended rectangles we check the aspect ratio and tolerance, to see if we need to stop extending. Rectangle R_{base} along with the extended rectangles will form the set $\mathcal{R}_{\text{left}}$. We then do the same extension operation on all rectangles in $\mathcal{R}_{\text{left}}$, but this time we extend rightwards. We have the same stopping criteria, checking for aspect ratio and tolerance of the extended rectangles. The newly extended rectangles, together with the rectangles in $\mathcal{R}_{\text{left}}$, form the set $\mathcal{R}_{\text{ext}}$.

We check for each rectangle $R \in \mathcal{R}_{\text{ext}}$, with label l_R and label aspect ratio a_{l_R}, whether the aspect ratio a_R is close enough to the aspect ratio a_{l_R}, by ensuring that $\rho_l \leq \frac{a_R}{a_{l_R}} \leq \rho_u$. If this is not the case, then we try to stretch R such that its aspect ratio gets closer to a_{l_R}, without covering any additional points. We do so by increasing the side length along either the major or minor axis (see Figure 8(a)). For example, assume the major axis of R is horizontal and $a_R < \rho_l \cdot a_{l_R}$. A similar procedure can be used when R is vertical or $a_R > \rho_u \cdot a_{l_R}$. We stretch R along its minor axis, creating up to three new candidates $\{R^u, R^d, R^c\}$: R^u stretches only upwards, R^d stretches only downwards, and R^c uses space above and below R (see Figure 8(b)). We stretch R until $a_R = \rho_l \cdot a_{l_R}$, or until we hit another point $p \notin R$; we never introduce new points into R since the resulting candidates will be handled by a different choice of p_i, p_j anyway. (If R is vertical those candidates would already be in $\mathcal{R}_{\text{ext}}$.)

We try to center R^c around R, using equal space above and below, but a point $p \notin R$ may be located nearby and this may force us to make adjustments. For example, if p is located above R, then we use all the available space above R, and as much as necessary below R, to ensure $a_R = \rho_l \cdot a_{l_R}$. However, if there is also a point too close below R, we cannot reach $a_R = \rho_l \cdot a_{l_R}$ without covering additional points. We remove R from $\mathcal{R}_{\text{ext}}$ and add the subset of $\{R^u, R^d, R^c\}$ of rectangles which pass the aspect ratio check and do not cover any additional points.

Next, we check for each rectangle $R \in \mathcal{R}_{\text{ext}}$ whether label l_R can be placed inside R at the minimum font size f. Remember that we always align the major and minor axis of a rectangle and

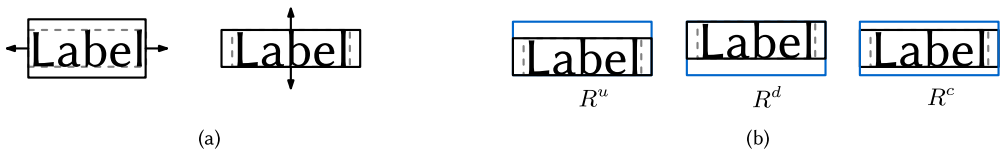


Fig. 8. Stretching R closer to the aspect ratio of the corresponding label l_R. (a) Stretching along major/minor axis depends on aspect ratio of R compared to l_R. (b) Three candidates for stretching: upwards, downwards, or both.

its assigned label. We remove R from $\mathcal{R}_{\text{ext}}$ if this check is not passed. The remaining rectangles form the candidate set $\mathcal{R}_{i,j}$.

Notice that a candidate set $\mathcal{R}_{i,j}$ does not produce any rectangle covering single points. To this end, we add candidates $\mathcal{R}_i$ for each point $p_i \in P$ by using the aforementioned stretching operation both horizontally and vertically from p_i. Since we created three options when stretching in one direction, there are nine possible outcomes when stretching both horizontally and vertically. The union of all candidate sets for pairs of points $\mathcal{R}_{i,j}$ and for individual points $\mathcal{R}_i$ forms the candidate set $\mathcal{R}$ that we use in our greedy heuristic and the SAT model.

$$\mathcal{R} := \bigcup_{\{p_i,p_j\}\subseteq P} \mathcal{R}_{i,j} \cup \bigcup_{p_i \in P} \mathcal{R}_i.$$

Given our candidate set $\mathcal{R}$ of rectangles that satisfy our constraints, we want to find a smallest subset $\mathcal{S} \subseteq \mathcal{R}$ of disjoint rectangles that covers all points in P or—if the minimum font size prevents some points from being covered—as many points in P as possible. Additionally, each candidate set $\mathcal{R}_{i,j}$ does not necessarily contain all viable rectangles, since we stop whenever the tolerance restriction is not met. In case we do not find all viable rectangles, even the smallest subset $\mathcal{S} \subseteq \mathcal{R}$ will not be an optimal solution for RPFA, but it is the best solution we can find heuristically.

4.2 Weighted Independent Set Model

In this section, we show how to find a good set of disjoint rectangles in the candidate set $\mathcal{R}$, by modeling our problem as the **Weighted Independent Set** (**WIS**) problem in the intersection graph on $\mathcal{R}$. An intersection graph on $\mathcal{R}$ has a vertex for each rectangle $R \in \mathcal{R}$ and an edge between two vertices if the corresponding rectangles intersect. A (weighted) independent set on this graph is a set of vertices, such that no two vertices share an edge, and hence no two rectangles corresponding to independent set vertices intersect. A benefit of this approach is that it allows us to utilize algorithmic approaches developed for this well-studied problem in previous works [1, 13]. In our setting, WIS takes as input a set of rectangles $\mathcal{R}$, and a weight function $w : \mathcal{R} \to \mathbb{R}^+$ which assigns a positive weight to each rectangle in R. We want to find a set $\mathcal{S} \subseteq \mathcal{R}$ of pairwise disjoint rectangles, so that their sum of weights is maximized.

Given an instance $\mathcal{I}$ of RPFA and a candidate set $\mathcal{R}$ consisting of all viable rectangles in the instance $\mathcal{I}$, we construct an instance $\mathcal{I}'$ of WIS by assigning weights to all candidate rectangles in $\mathcal{R}$ as follows. Let R be a rectangle in the candidate set, and $|R|$ denote the number of points that lie inside R. Furthermore, let $n = |P|$ be the number of points in $\mathcal{I}$. For the instance $\mathcal{I}'$ we take the candidate rectangles $\mathcal{R}$, and assign to each rectangle $R \in \mathcal{R}$ the weight $2n|R| - 1$. We say that $\mathcal{I}'$ is the corresponding WIS instance of $\mathcal{I}$.

Let $\mathcal{S}' \subset \mathcal{R}$ be a valid independent set, i.e., the set $\mathcal{S}'$ consists of pairwise non-overlapping rectangles. Then, in the instance $\mathcal{I}'$, the weight of the set $\mathcal{S}'$ is $W(\mathcal{S}') = \sum_{R\in\mathcal{S}'}(2n|R| - 1)$. Let $P_{\mathcal{S}'}$ be the point set consisting of points in P that are covered by rectangles in $\mathcal{S}'$. Since all rectangles

in $\mathcal{S}'$ are disjoint, we can rewrite the above equality to express the total weight in terms of $|P_{\mathcal{S}'}|$ and $|\mathcal{S}'|$ and get $W(\mathcal{S}') = 2n|P_{\mathcal{S}'}| - |\mathcal{S}'|$.

We now prove that, using these weights, a maximum weight independent set in $\mathcal{I}'$ corresponds to a set of disjoint rectangles in $\mathcal{I}$ which maximizes the number of covered points, and among all such sets is of minimum size.

Lemma 4.1. *Let $\mathcal{S}, \mathcal{S}' \subseteq \mathcal{R}$ each be an independent set of rectangles in a WIS instance $\mathcal{I}'$ reduced from RPFA instance $\mathcal{I}$, such that $\mathcal{S}$ covers more points than $\mathcal{S}'$, i.e., $|P_{\mathcal{S}}| > |P_{\mathcal{S}'}|$. Then $\mathcal{S}$ has a larger weight than $\mathcal{S}'$, i.e., $W(\mathcal{S}) > W(\mathcal{S}')$.*

Proof. Since $\mathcal{S}$ and $\mathcal{S}'$ both consist of pairwise non-overlapping rectangles, each point in P can be covered by at most one rectangle in $\mathcal{S}$ and also in $\mathcal{S}'$. Then by the pigeon hole principle, $\max(|\mathcal{S}|, |\mathcal{S}'|) \leq n$. Thus:

$$W(\mathcal{S}) = 2n|P_{\mathcal{S}}| - |\mathcal{S}| > 2n(|P_{\mathcal{S}}| - 1) \geq 2n|P_{\mathcal{S}'}| \geq 2n|P_{\mathcal{S}'}| - |\mathcal{S}'| = W(\mathcal{S}').$$ □

Theorem 4.2. *Let $\mathcal{I}$ be an instance of RPFA, and $\mathcal{I}'$ be the corresponding* WIS *instance. Then an optimal solution $\mathcal{S}^*$ of $\mathcal{I}$ corresponds to a maximum weight independent set in $\mathcal{I}'$.*

Proof. We make a case distinction on whether or not $\mathcal{I}$ has a non-zero minimum font size f.

Minimum font size $f = 0$: In this case, $\mathcal{S}^*$ must cover all points in P. That means $P_{\mathcal{S}^*} = P$, and also implies $W(\mathcal{S}^*) = 2n|P| - |\mathcal{S}^*| = 2n^2 - |\mathcal{S}^*|$. First, observe that $\mathcal{S}^*$ corresponds to an independent set by definition, and hence we only have to argue that $\mathcal{S}^*$ corresponds to an independent set that maximizes the weight. Let $\mathcal{S}$ correspond to an independent set in $\mathcal{I}'$. If $\mathcal{S}$ does not cover all points, then by Lemma 4.1 the weight of $\mathcal{S}$ cannot be larger than $W(\mathcal{S}^*)$. Thus assume that $\mathcal{S}$ covers all points in P, and assume for contradiction that $W(\mathcal{S}) > W(\mathcal{S}^*)$. We get

$$2n^2 - |\mathcal{S}| = W(\mathcal{S}) > W(\mathcal{S}^*) = 2n^2 - |\mathcal{S}^*|.$$

Hence it must hold that $|\mathcal{S}| < |\mathcal{S}^*|$. This contradicts the fact that $\mathcal{S}^*$ is an optimal solution for $\mathcal{I}$, since $\mathcal{S}$ also covers all points, is independent, and consists of fewer rectangles.

Minimum font size $f > 0$: In this case, an optimal solution $\mathcal{S}^*$ need not cover all points in P. However, $\mathcal{S}^*$ covers as many points as possible. That is, for any independent set $\mathcal{S}$ in $\mathcal{R}$, we have $|P_{\mathcal{S}}| \leq |P_{\mathcal{S}^*}|$. If $\mathcal{S}$ covers fewer points than $\mathcal{S}^*$, then by Lemma 4.1, the weight of $\mathcal{S}$ is smaller than the weight of $\mathcal{S}^*$. Since $\mathcal{S}^*$ is an independent set by definition, there must be an independent set $\mathcal{S}$ that covers at least as many points as $\mathcal{S}^*$. We can hence repeat the argument in the previous case, showing that there is no solution $\mathcal{S}$ for $\mathcal{I}'$ that has higher weight than $\mathcal{S}^*$, and thus fewer rectangles. □

Lemma 4.3. *Let $\mathcal{I}$ be an instance of RPFA, and $\mathcal{I}'$ be the corresponding* WIS *instance. Then each maximum weight independent set in instance $\mathcal{I}'$ is an optimal solution $\mathcal{S}^*$ in $\mathcal{I}$.*

Proof. Consider a maximum independent set $\mathcal{S}$ in the instance $\mathcal{I}'$. By Lemma 4.1, we know that $\mathcal{S}$ must cover as many points as possible. Furthermore, there cannot be an independent set $\mathcal{S}'$ in $\mathcal{I}'$, that covers as many points as $\mathcal{S}$ ($|P_{\mathcal{S}}| = |P_{\mathcal{S}'}|$), but has fewer rectangles ($|\mathcal{S}| > |\mathcal{S}'|$). For such an independent set $\mathcal{S}'$ the inequality $W(\mathcal{S}') = 2n|P_{\mathcal{S}'}| - |\mathcal{S}| > 2n|P_{\mathcal{S}}| - |\mathcal{S}| = W(\mathcal{S})$ would hold, contradicting the optimality of $\mathcal{S}$. □

4.3 Exact Solver Based on MAXSAT

Recently, Klute et al. [23] proposed an exact MAXSAT model for the point feature labeling problem. They achieved reasonable running time for real-world cartographic applications. Using a similar idea, we propose an exact approach based on a SAT model, modeling weighted independent set as

a set of clauses and Boolean variables. We did preliminary tests with ILP and CSP approaches, but we ultimately found that using a SAT model was the most scalable.

Satisfiability. A clause c is a disjunction of a set of Boolean variables, where each variable v appears either as a positive literal v or a negative literal $\neg v$ in c. Given a variable set V, a truth assignment $\alpha : V \to \{true, false\}$ maps a truth value to each variable in V. Given a truth assignment, one evaluates a clause c as *true* if and only if at least one literal in c is *true*. A *weighted* SAT formula ϕ in conjunctive normal form is a conjunction of clauses $\phi = c_1 \wedge \cdots \wedge c_m$, where each clause c_i is assigned a positive weight $w(c_i)$. The weighted maximum satisfiability problem (Weighted MAXSAT) asks for a truth assignment α that maximizes the total weight of the satisfied clauses in ϕ.

We now encode an instance $\mathcal{I}$ of the WIS problem, consisting of the set $\mathcal{R}$ of weighted candidate rectangles, in a weighted SAT formula $\phi_{\mathcal{I}}$ as follows. For each rectangle $R \in \mathcal{R}$, we introduce a variable v_R. We then construct two types of clauses. For each rectangle R, we introduce a rectangle clause $c_R = v_R$ consisting of the single positive literal v_R. The weight of c_R is the weight of the corresponding rectangle R, namely $w(c_R) = 2n|R| - 1$. For each intersection between two rectangles R and R', we introduce an intersection clause $c_{R,R'} = \neg v_R \vee \neg v_{R'}$ with weight $w(c_{R,R'}) = 2|\mathcal{R}|n^2$.

Given a truth assignment α, the set of *true* variables corresponds to a set $\mathcal{S}_\alpha \subseteq \mathcal{R}$ of rectangles. Conversely, given a set $\mathcal{S} \subseteq \mathcal{R}$, the corresponding truth assignment $\alpha_{\mathcal{S}}$ assigns *true* to a variable v_R if and only if $R \in \mathcal{S}$.

Observation 1. *A set $\mathcal{S} \subseteq \mathcal{R}$ is an independent set if and only if its corresponding truth assignment $\alpha_{\mathcal{S}}$ satisfies all intersection clauses.*

Given a truth assignment $\alpha_{\max}$ that maximizes the total weight of satisfied clauses in $\phi_{\mathcal{I}}$, we prove the following.

Lemma 4.4. *Every intersection clause is satisfied in α_{max}.*

Proof. Given a rectangle R, its rectangle clause c_R has weight $2n|R| - 1 < 2n^2$. Thus the total weight of all rectangle clauses is bounded by the weight $2|\mathcal{R}|n^2$ of a single intersection clause. Consider the special truth assignment α_{false} that assigns *false* to each variable. Using this special assignment α_{false}, each intersection clause is satisfied. Now suppose, for contradiction, that there exists an intersection clause that evaluates to *false* by $\alpha_{\max}$. Then the total weight of *true* clauses in the assignment $\alpha_{\max}$ is less than the total weight of α_{false}, even in the case that $\alpha_{\max}$ satisfies all rectangle clauses. This contradicts the assumption that $\alpha_{\max}$ maximizes the total weight. □

Theorem 4.5. *Let α be a truth assignment of $\phi_{\mathcal{I}}$ and $\mathcal{S}_\alpha$ be the rectangle set corresponding to all variables that are assigned true in α. Assignment α maximizes the total weight of satisfied clauses if and only if the set $\mathcal{S}_\alpha$ is a maximum weight independent set in $\mathcal{R}$.*

Proof. Suppose $\mathcal{S}_\alpha$ is a maximum weight independent set, then by Observation 1 its corresponding assignment α must satisfy all intersection clauses. By optimality of $\mathcal{S}_\alpha$, the total weight of satisfied rectangle clauses is also maximized by α, and hence α maximizes the overall weight of satisfied clauses.

Conversely, assume that α is an assignment that maximizes the weight of all satisfied clauses. The corresponding rectangle set $\mathcal{S}_\alpha$ is an independent set, by Observation 1 and Lemma 4.4. Furthermore, as α satisfies all intersection clauses and maximizes the total weight of satisfied rectangle clauses, $\mathcal{S}_\alpha$ has the maximum weight among all independent sets. □

After we encode our problem as a weighted MAXSAT problem, we use the open-source MAXSAT solver MaxHS 3.0 [17].

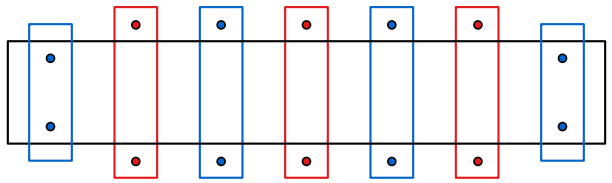


Fig. 9. A point set with corresponding candidate set, consisting of 7 vertically aligned candidate rectangles (blue/red), 1 horizontally rectangle (black), and singletons (not drawn).

4.4 Greedy Heuristic

Since the running time of the exact MAXSAT solver is infeasible for larger instances (see Section 5), we propose a simple heuristic to find solutions for WIS more quickly. We first assign weights to each rectangle in $\mathcal{R}$, as explained in Section 4.2, to ensure that an optimal solution for WIS corresponds to an optimal solution for RPFA. The heuristic starts by choosing the rectangle in $\mathcal{R}$ that has the highest weight as the initial solution $\mathcal{S}$. We then consider the remaining rectangles in order of descending weight, and greedily try to add them to $\mathcal{S}$, one by one. A rectangle $R \in \mathcal{R}$ can either overlap with a rectangle in $\mathcal{S}$, or it is completely disjoint from all rectangles in $\mathcal{S}$. In the former case, we discard R, while in the latter case, R is added to $\mathcal{S}$. We proceed until all input points P are covered, or otherwise until $\mathcal{R}$ has been exhausted. Recall that not all rectangles covering single points may be present due to the constraints, and even when they are, we may have chosen rectangles in $\mathcal{S}$ that overlap with these singleton rectangles. Hence there are inputs for which we exhaust $\mathcal{R}$ before all points are covered.

4.5 Comparison of MaxSAT and Greedy

In the following, we give a comparison of these two approaches with a simple example. We presented an exact solver based on MAXSAT (see Section 4.3) and a greedy heuristic (see Section 4.4) for the RPFA problem. Both approaches take the same set of candidate rectangles as input and solve the WIS problem, either exactly or heuristically. We illustrate these two procedures and their differences with an example instance; see Figure 9.

In this example, the candidate rectangle set $\mathcal{R}$ consists of seven vertically stretched rectangles, one horizontal rectangle, and singletons for each point. Recall that we assign to each candidate rectangle $R \in \mathcal{R}$ the weight $2n|R| - 1$, where $n = |P|$ and $|R|$ is the number of points covered by R. The maximum weight solution $\mathcal{S}^*$ then consists of the seven vertically aligned rectangles. Our exact solver method encodes these weights and pairwise overlaps of candidate rectangles as a weighted MAXSAT problem and then solves this problem exactly, thus finding the solution $\mathcal{S}^*$.

Our greedy heuristic sorts the candidate rectangles in $\mathcal{R}$ by their weights in decreasing order. Then a set $\mathcal{S}$ of pairwise disjoint rectangles is built greedily by choosing the rectangles in this order. Precisely, this heuristic starts with the highest-weight rectangle R_1 and adds it to $\mathcal{S}$. Then it checks the highest-weight rectangle R_2 in $\mathcal{R} \setminus \{R_1\}$. If this rectangle R_2 is entirely disjoint from all rectangles in $\mathcal{S}$, we add it greedily to $\mathcal{S}$; otherwise it is discarded. This process is repeated until all points are covered by $\mathcal{S}$ or the candidate set $\mathcal{R}$ is exhausted.

The horizontal rectangle H, which covers 4 blue points, has the highest weight. Thus, the greedy approach collects H as the first rectangle of the greedy solution $\mathcal{S}$. All vertical candidate rectangles intersect H. Thus, after picking H, an uncovered point can only be covered by its singleton. Hence, the greedy solution $\mathcal{S}$ consists of H and ten singletons for the points outside H. Extending this simple construction by adding more vertical rectangles with appropriately colored points, the greedy approach could in principle be far from optimal in the worst case. However, we did not observe such behavior in our experiments.

5 EXPERIMENTAL EVALUATION

We implemented the algorithms in Section 4 and first compared them against each other on synthetic data, e.g., showing that our heuristic scales well and finds solutions that are close in quality to the exact model in most cases. In the second part of the evaluation, we provide a case study in which we showcase how Worbels can be used on a real-world data set.

5.1 Experimental Setup

The experiments for measuring performance were run on a server equipped with two Intel Xeon E5-2640 v4 processors (2.4 GHz 10-core) and 160 GB RAM, operating the 64-bit version of Ubuntu Bionic (18.04.2 LTS). The case study was carried out on a standard Windows laptop equipped with an Intel Core i7-6700HQ (2.60 GHz, 4 Cores 8 Logical Processors) and 16 GB RAM. The code was compiled with g++ 7.4.0 with Optimization level -O3.

Data sets. We use three types of data for our experiments: two synthetic data sets that we use to analyze the scalability of our algorithms and the quality of their output, and a real-world data set to show the viability of our technique in practice. The synthetic data sets, Uniform and Gaussian, are randomly generated point sets inside a bounding box of size 1000×1000 pixels, based on a uniform and a Gaussian distribution, respectively. The k category labels in these synthetic instances are generated randomly such that each label has 3 to 10 (lowercase) letters. The number of characters is chosen uniformly at random for each of the labels. We vary the number of points n (from 20 to 1,000, with increments of 10), as well as the number of categories k (2, 3, 4, 8, or 16 categories).

Uniform. In the uniform model, we generate n points inside a bounding box $\mathcal{B}$ of size 1000×1000 pixels uniformly at random. Then we assign one of the k categories to each point uniformly at random.

Gaussian. Intuitively, the points in a Gaussian instance are drawn from a mixture of k Gaussian distributions. We first decide the number of points in each category $\{n_1, \ldots, n_k\}$, using a Dirichlet distribution with dimension k. Note that the numbers in the categories sum up to n. For each category i, we use a separate Gaussian distribution whose mean μ_i is sampled in the bounding box $\mathcal{B}$ uniformly at random. Then a value σ_i is sampled uniformly between 0 and 0.5, and is used as the standard deviation for both dimensions. We then generate, for each $i = 1, \ldots, k$, n_i points inside $\mathcal{B}$ using a Gaussian distribution with parameters μ_i and σ_i.

Every five years, the **U.S. Department of Agriculture** (**USDA**) publishes a comprehensive summary of the census of agriculture, which includes harvest statistics for major crops in each county. For the real-world data, US-Crops, we use statistical data about the crop harvest in the United States in the year 2007.[2] The data is on county level, resulting in 3,067 data points, with a label from the set {Corn, Wheat, Cotton, Soybeans, Vegetables, Various} indicating the dominant crop in the harvest of each county. Additionally, we filter the counties with at least 100 000 acres harvested, to get a smaller data set with 817 points, which we call US-Crops-filtered, and a variant of the data set where the aggregation is on state level, US-Crops-state, resulting in 48 data points. The point feature for each county/state uses the latitude and longitude of a central location in each county/state as coordinates: We hence use a equirectangular projection for our data points, by using (lat, long) as coordinates, as well as for our background map.[3] However, any other projection should work as well, as long as our algorithms are run on the projected points, to ensure a grouping of projected points into rectangular label areas in the visualization.

[2] extracted from ArcGIS_US_CROPS_2007.
[3] https://commons.wikimedia.org/wiki/File:USA_location_map.svg.

Real-world. We generate three data sets from this data base, where we assign the labels representing the dominant crop to each county/state.

US-Crops. The full data extracted from the data base. Each point feature represents a county in the data. This data set consists of 3067 counties, partitioned into the following six categories: {Corn, Wheat, Cotton, Soybeans, Vegetables, Various}. The label Various is assigned, if there is no data for that county in the census; any crop may be dominant.

US-Crops-filtered. In this middle-size data set, we filter all counties with total area of harvested crops larger than 100 000 acres, resulting in a total of 817 features with five categories, all previous categories, minus the label Various, since those states will always be filtered out.

US-Crops-state. In this state-level data set, each point feature represents one state. We compute the dominant crop per state by summing up acres of the counties in a state, and assigning the crop with the highest total harvest as the dominant crop in the state. Then, we use the latitude and longitude obtained from the Geographic Coordinates of US States as coordinates for a point feature representing the state. This instance has 48 points and each point is assigned one of five labels, similar to the filtered data set.

5.2 Metrics

In this section, we introduce the metrics used in our evaluation to measure the quality of the produced solutions. Each metric aims at measuring a different aspect of the solution quality, and relates to the constraints introduced in Section 2.

Solution Size [SIZE]. Our primary indicator for how well our solution aggregates data (and resembles a word cloud) is the number of rectangles in a solution $\mathcal{S}$. If we simply have one rectangle per input point, we are labeling the individual points. As the number of rectangles decreases, the number of covered points per rectangle increases, and a solution will aggregate more data in a single rectangle, which will be represented by a single word in the visualization.

Uncovered Points [MISS]. Since our visualization aims at showing an at-a-glance overview of the categorical point data, we set a minimum font size for the words in the visualization, to prevent the necessity to zoom and pan. However, this lower bound may prevent some points from being covered, as there may be no rectangle that both covers a point, and fits the category label at the desired font size, without overlapping other rectangles in the solution. Hence we measure how often this undesired behavior occurs. Though, some uncovered points may be close to a rectangle that represents their category, and therefore will still be well-represented in the visualization. To capture the severity of not covering an input point, we also measure the distance from an uncovered point to a rectangle representing their category.

Specifically, these two variants of the MISS metric are defined as follows. Let P_U be the set of uncovered points in a solution $\mathcal{S}$. Then the metric $\text{MISS}_\#$ captures the number of uncovered points.

$$\text{MISS}_\# := |P_U|.$$

Additionally, let $H(p, R)$ be the Hausdorff distance between a point p and a rectangle R. If point p is inside R, then this distance will be 0, otherwise the distance is measured as the smallest Euclidean distance between p and R. For an uncovered point $p \in P_U$, let $\ell(p)$ be the label of p, and let $\mathcal{S}_{\ell(p)}$ be the set of rectangles for label $\ell(p)$ in solution $\mathcal{S}$. For every $p \in P_U$, we measure the minimum Hausdorff distance to any rectangle $R \in \mathcal{S}_{\ell(p)}$. Similar to Measure 1 in [11], we define metric MISS_H as the sum over these distances, for all points in P_U.

$$\text{MISS}_H := \sum_{p \in P_U} \min_{R \in \mathcal{S}_{\ell(p)}} H(p, R).$$

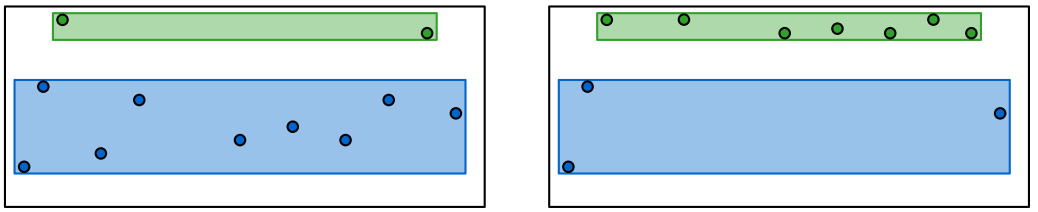


Fig. 10. Density variance. Examples with (a) low density variance and (b) high density variance.

Text Coverage [TEXT]. The text labels for each category have a fixed aspect ratio, since we only vary the font size, as opposed to scaling the labels arbitrarily. As a result, the text in the visualization may not completely fill the rectangles they represent. To measure how well the text labels cover the points inside the rectangles of a computed solution $\mathcal{S}$, we introduce the TEXT metric, with a counting and a distance-based variant $\text{TEXT}_\#$ and TEXT_H, respectively.

Let P_R be the points covered by rectangle R, and $P_{T(R)} \subseteq P_R$ the points inside the bounding box $T(R)$ of the text label placed in R. We then define $\text{TEXT}_\#$ as follows.

$$\text{TEXT}_\# := \sum_{R \in \mathcal{S}} |P_R \setminus P_{T(R)}| = \sum_{R \in \mathcal{S}} (|P_R| - |P_{T(R)}|).$$

Additionally, we observe that a point $p \in P_R \setminus P_{T(R)}$, which is located close to $T(R)$ is better represented, than a point further removed from $T(R)$. To measure the total error in a solution, we take the sum of the Hausdorff distances between every point $p \in P_R \setminus P_{T(R)}$ to $T(R)$, for all rectangles in $\mathcal{S}$.

$$\text{TEXT}_H := \sum_{R \in \mathcal{S}} \sum_{p \in P_R \setminus P_{T(R)}} H(p, T(R)).$$

Density [DENS]. Since we are not labeling individual points, but placing text labels that cover an area, users may interpret such labels as areas, which have data points spread all throughout. It may happen, however, that candidate rectangles cover a large area, while containing only a few data points. Such labels do not represent the data well, and hence we prefer labels that cover an area that contains points more uniformly. We measure this relation between the number of covered points $|P_R|$ and the covered area $A(R)$, as the fraction of these two values.

$$\text{DENS}_R := |P_R|/A(R).$$

For a solution $\mathcal{S}$ of disjoint rectangles, we evaluate the variance in the densities DENS_R for all $R \in \mathcal{S}$. If the variance is low, then all rectangles have about the same density, and hence larger rectangles cover a larger amount of points. This is a characteristic seen in word clouds, where words are scaled by importance or number of occurrences. For high variance, this relation is lost, and hence the solution does not exhibit this behavior; see Figure 10.

Normalization. The metrics introduced above do not compare well between different data sets, if the characteristics of the data, such as number of points or the spread, differ a lot. We therefore introduce a normalization for most metrics, to make the results more comparable between data sets.

For the # variants of our metrics, we divide by $|P|$, to get a ratio of the total number of points. Furthermore, let the coordinates of input points P vary between $[0, x_{\max}]$ and $[0, y_{\max}]$, and let $d = \sqrt{x_{\max}^2 + y_{\max}^2}$. We normalize the H variants of our metrics by dividing by d, following the

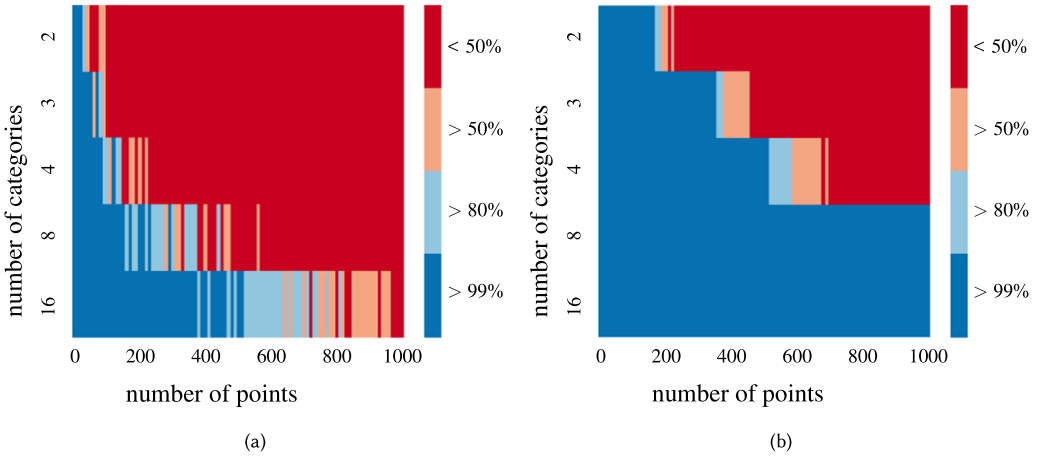


Fig. 11. Percentage of instances, for which the exact solver terminates within 30 minutes. (a) Gaussian, and (b) Uniform instances with $n = 20$ to 1,000 points, $c \in \{2, 3, 4, 8, 16\}$ categories.

evaluation in [11]. This measures the sum of errors relative to a diagonal of the bounding box of P, which is a rough indicator of distances in a data set.

5.3 Experimental Results

Here we explain the results from our experiments, starting with the comparison between our two algorithms.

5.3.1 Exact Solver vs Greedy Heuristic. For our first set of experiments, we compare the exact solver and the greedy heuristic. In this experiment, both algorithms were executed with parameter settings $\rho_l = 0$, $\rho_u = \infty$, and $t = f = 0$.

— **Instance Parameters and Performance.** We first show that the running time of both the exact solver and the greedy heuristic are greatly affected by the size of the candidate set $\mathcal{R}$, more so than by the number of points n or number of categories c. Figure 11 shows the percentage of instances of Gaussian and Uniform data sets for which the MAXSAT solver terminates within 30 minutes, for all combinations of n and c. As n grows, we see that the percentage of failed runs grows as well, but not very strongly when c is large ($c = 16$ for Gaussian, $c = 8$ and up for Uniform). However, when c is smaller, especially $c = 2$, less than half of the instances finish within half an hour, even for smaller point sets, such as $n = 300$. To understand this discrepancy, we turn to Figure 12.

Figure 12(a) shows the relation between the size of the candidate set, and the running time, for each individual Gaussian instance. The y-axis ends at a running time threshold of 30 minutes, and we plot a red point at 30 minutes when a run did not finish. This happens only for the exact solver, with the exception of a single run for the greedy approach that ran out of memory. As the candidate set grows, we see the running time of the exact solver growing much more quickly than the running time of the greedy heuristic: while the greedy approach can still find a solution in under a second for most instances where the candidate set consists of around 100 000 rectangles, the exact solver cannot find a solution within 30 minutes in almost all cases even for 6,500 rectangles. We stopped running the exact solver when candidate sets exceed 100 000 rectangles. Note that the 30 minute timeout was chosen for practical reasons only, and many instances which produce more than 5,000 rectangles

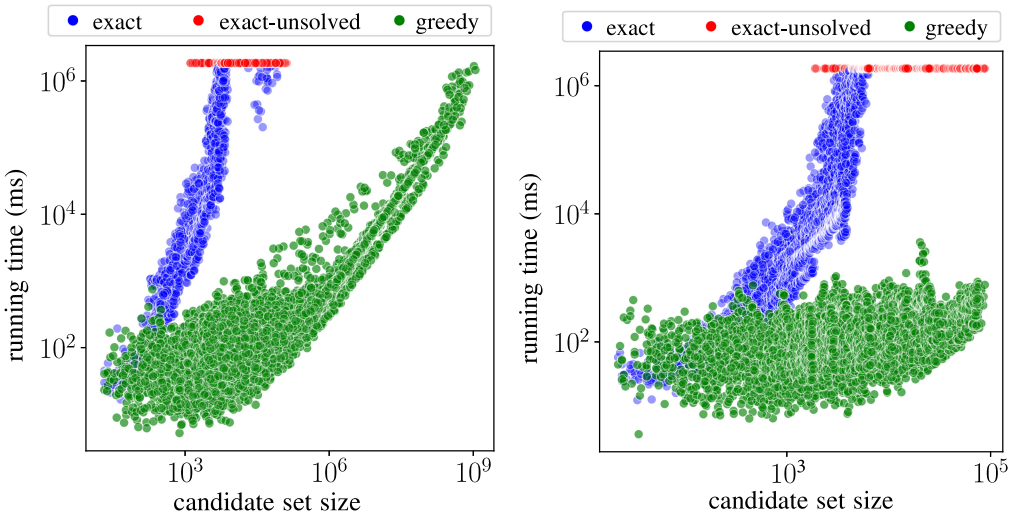


Fig. 12. Log-log plot of running time with respect to candidate set size of (a) Gaussian and (b) Uniform instances. Red points show instances that failed to finish in 30 minutes, for candidate sets up to 10^5. On a vertical line, red and blue points sum up to an equal number of green points.

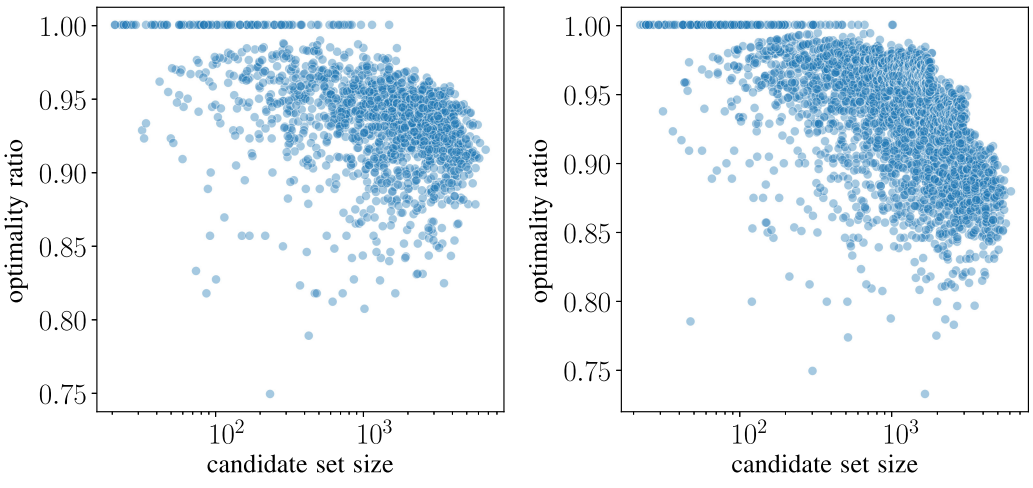
in the candidate set may take much longer than 30 minutes to finish using the exact solver. This prevents the exact solver from being a viable option for all but the smallest instances in practice. On the other hand, the greedy heuristic can still find solutions within 30 minutes, even for candidate sets of up to 100 million rectangles.

The results for the Uniform data sets are very similar; see Figure 12(b). The running time of the greedy heuristic grows very slowly as the candidate set size increases. On the other hand, the exact solver again shows a steep increase in running time, as we also saw for Gaussian instances (Figure 12(a)). Similar as for the Gaussian instances, the greedy heuristic shows a significantly faster running time, already for small instance size of 1,000 candidate rectangles, while the heuristic also solves large instances even with up to 100 000 rectangles.

We believe that the growth of the candidate set, for high values of n and especially for small values of c, can be explained as follows. During the computation of the candidate set, fewer categories lead to many strips with base rectangles that can be extended often, because the probability of encountering a point of a different color is lower. These cases should occur even more often in Gaussian than in Uniform, and we indeed find that the number of instances that finishes in 30 minutes is lower for Gaussian, judging by Figure 11. Additionally, we can confirm that the candidate sets for Uniform instances never exceed 100 000 rectangles.

— **Optimality Gaps.** Now that we have established that the exact solver, unlike the greedy heuristic, is not viable for larger instances, we proceed to verifying whether the heuristic finds near-optimal solutions. We therefore measure the ratio between the number of rectangles in a solution produced by the greedy heuristic and an optimal solution for a given candidate set. The optimal solution size is found by the exact solver, and hence we can only do this analysis for data sets where the exact solver actually terminates.

In Figure 13(a) we show this ratio for all Gaussian instances where the exact solver found a solution. We see that the solutions found by the greedy approach deviate more



(a) Gaussian (b) Uniform

Fig. 13. Ratio of the solution size between greedy and optimal solutions for (a) Gaussian and (b) Uniform instances with various candidate set sizes.

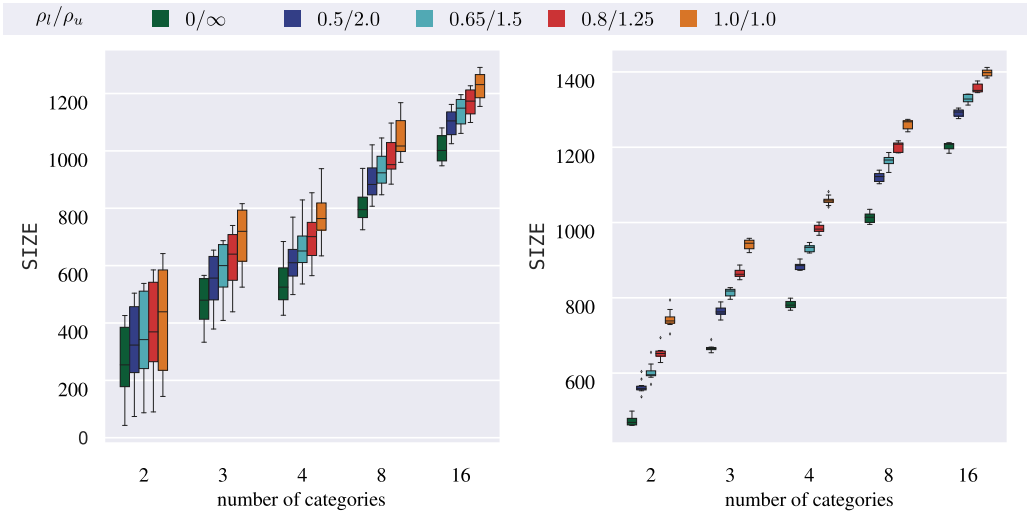
from optimality, as the candidate set size increases. However, for the majority of evaluated instances, the solution size computed by the greedy heuristic is at most 10% larger than the optimal solution, and there are only two instances where this difference grows to 20%.

Just as for the Gaussian instances, we consider only the small Uniform instances, for which the exact solver found a solution. The plot (Figure 13(b)) shows that the greedy heuristic consistently finds solution sizes higher than 70% of the optimum solution size. For most of the evaluated instances, the size of the greedy solution is at least 85% of the optimum solution size. For small instances with at most 100 candidate rectangles, the greedy solutions reach the optimum sizes for most of the instances. As expected, the optimality ratios are distributed more widely as the candidate set size increases. This again mirrors the same findings for the Gaussian instances.

We also see that the greedy approach is able to find optimal solutions for some instances. This means that our greedy heuristic is not only much faster than the exact solver, but also produces solutions that are still of high quality in terms of solution size. We therefore focus on the solutions produced by the greedy heuristic in our case study. Overall, the performance of the greedy heuristic shows a well-balanced compromise between running time and solution quality on these instances.

5.3.2 Parameters and Metrics. For the second set of experiments, we compare variants of the greedy heuristic with different parameter settings and investigate how these parameters affect the solution quality in our defined metrics. Here we explain the results using the synthetic data. For each of Gaussian and Uniform model, we generate 10 random instances with 1,600 points of k colors for $k \in \{2, 3, 4, 8, 16\}$ and measure the metrics presented in Section 5.2

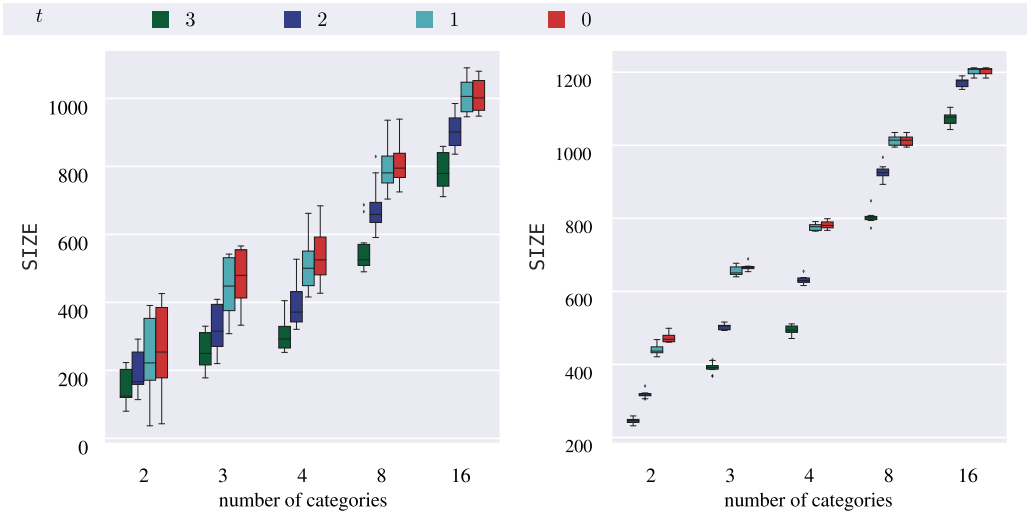
— **Parameters and Solution Size.** We first compare variants of greedy heuristic with various parameter settings in terms of solution size and show the results in Figure 14, Figure 15 and Figure 16. These variants show their predicted behavior: For the parameters ρ_l and ρ_u, the more restricted the allowed aspect ratio of the cover rectangles, the greater the solution size (Figure 14(a) and (b)); Similarly, for tolerance threshold t, the less tolerance for



(a) Gaussian, aspect ratio.

(b) Uniform, aspect ratio.

Fig. 14. Box plot of solution size and aspect ratio, grouped in number of categories. Each box shows the results with specific aspect ratio bounds, on 10 instances.

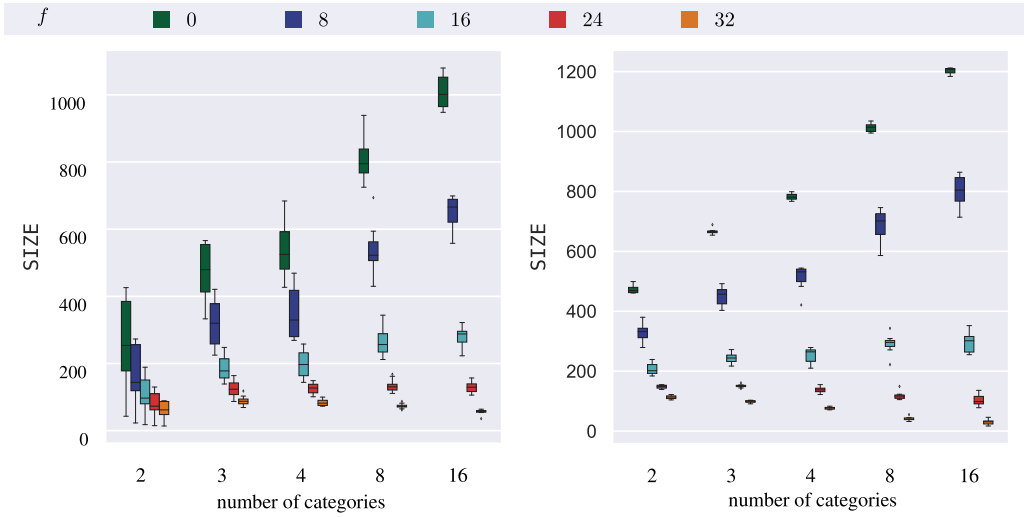


(a) Gaussian, misrepresentation tolerance.

(b) Uniform, misrepresentation tolerance.

Fig. 15. Box plot of solution size and misrepresentation tolerance, grouped in number of categories. Each box shows the results with specific tolerance threshold, on 10 instances.

misrepresentation, the more cover rectangles are required (Figure 15(a) and (b)); Recall that it may not be possible to cover all points with the constraint on the minimum font size; and the aim is to place a maximal number of viable rectangles. Here, the greater the minimum font size, the smaller the solution size (Figure 16(a) and (b)). The results for Gaussian and Uniform data sets show similar patterns, except that the results of Uniform data have less variance in the measured values due to the similarity of large uniform instances. As the number of categories grows, we see that the solution size increases (decreases in the plots for the



(a) Gaussian, minimum font size.

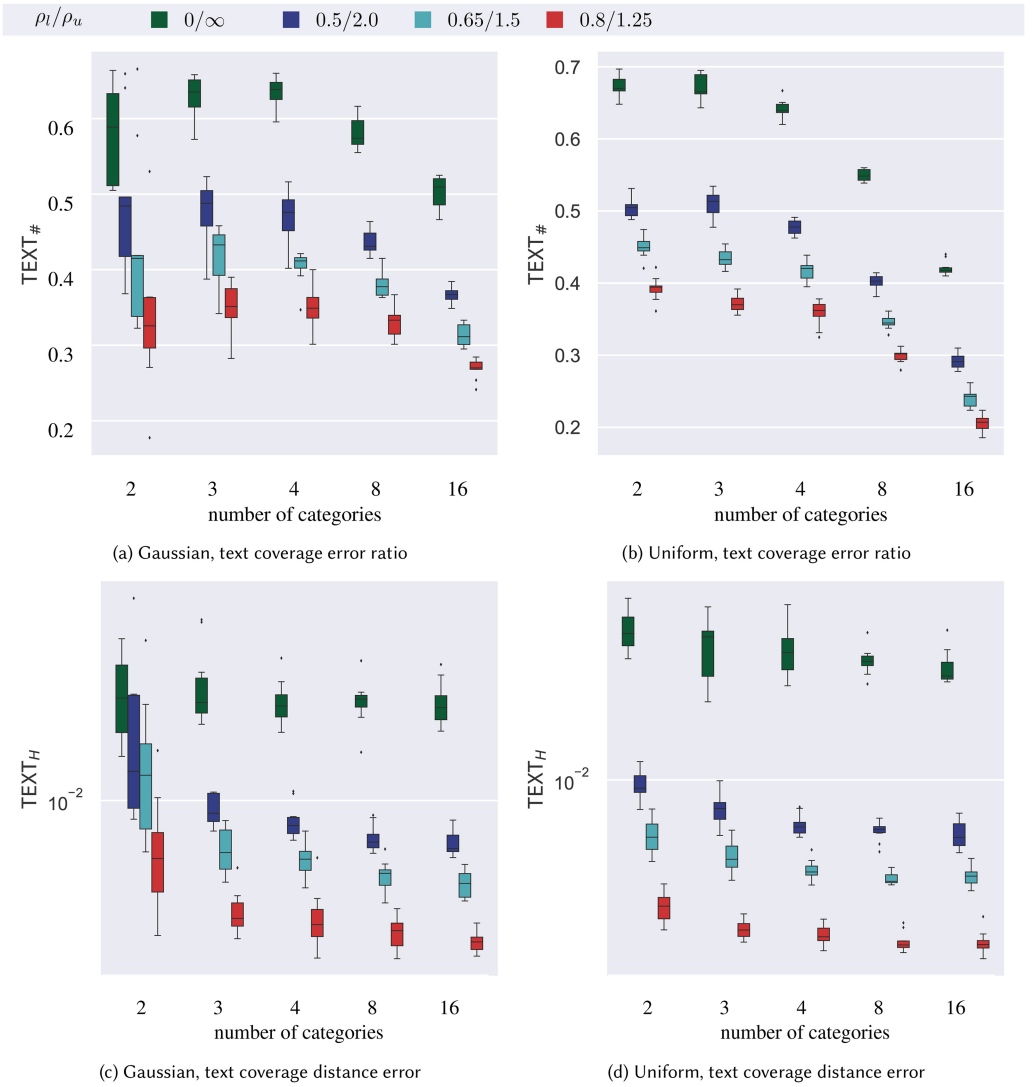
(b) Uniform, minimum font size.

Fig. 16. Box plot of solution size and minimum font size, grouped in number of categories. Each box shows the results with a specific minimum font size, on 10 instances.

minimum font size). We believe this is quite natural since the more categories are present, the more likely a neighboring point is of a different category.

— **Aspect Ratio vs Text Coverage.** Next, we explore in more detail the average text coverage of the computed solution, relative to aspect ratio constraint. Here we measure two metrics: $\text{TEXT}_{\#}$ for the ratio of points that are not covered by the bounding box of the corresponding label and TEXT_H for the average distance of a point to its corresponding label bounding box, normalized by dividing the diagonal of the corresponding label bounding box. The result of the Gaussian and Uniform data model are plotted in Figure 17. An expected observation is that, as the allowed aspect ratio of rectangles gets closer to the actual bounding box of the label ($\rho_l = \rho_u = 1.0$), more points are covered by the bounding box of the text (Figure 17(a) and (b)). It shows that our proposed parameters for the aspect ratio constraint enable a flexible control of text coverage, thus provide a powerful customization tool for the final visualization. We can observe that in most cases at most 50% of the points in a rectangle are not covered by the corresponding text. Those uncovered points are close to the bounding box of the text (under 1% of its diagonal length) in the most cases (Figure 17(c) and (d)).

— **Minimum Font Size and Uncovered Points.** We explore the degree to which points are left uncovered with various minimum font sizes. Recall that with the constraint of minimum font size, a valid solution covering all points might not exist. In such setting, the goal is to cover as many points as possible. Here we measure two metrics for uncovered points: $\text{MISS}_{\#}$ for the ratio of uncovered points in a solution and MISS_H for the average distance of uncovered points to its corresponding cover rectangle, normalized by dividing by the diagonal length of their corresponding cover rectangles. The results are plotted in Figure 18. We consider first the ratio of uncovered points for Gaussian instances (Figure 18(a)) and Uniform instances (Figure 18(b)). We observe that the number of uncovered points increases as the minimum font size increases, except for uniform instances of 16 categories. Recall that we compare the variants of our greedy heuristic, which can explain this irregularity. With more restrictions some chosen rectangles may no longer be candidates and this may lead to better solutions.



(a) Gaussian, text coverage error ratio

(b) Uniform, text coverage error ratio

(c) Gaussian, text coverage distance error

(d) Uniform, text coverage distance error

Fig. 17. Box plot of text coverage error with respect to number of categories of Gaussian (left) and Uniform instances (right). Each box shows the results of the greedy heuristic with specific aspect ratio bounds, on 10 instances. The text coverage distance error of each point is normalized by dividing by the diagonal length of its corresponding cover rectangle.

It is interesting to see that, as the minimum font size grows, the ratio of uncovered points increases dramatically, while the distance of the uncovered points to its closest cover rectangle increases slower (Figure 18(c) and (d)); It is important to note that the y-axis of distance plot is in log scale. Moreover, even if the ratio of uncovered points is usually more than 50% when we insist on large font sizes and a large number of categories, the distance of an uncovered point to the closest cover rectangle of the same category is under $\frac{1}{10}$ of the diagonal length of the cover box in the most of cases.

— **Parameters and Density.** Finally, we explore the metric density of the computed solution, which measures the variance of point density of cover rectangles. Since word clouds often

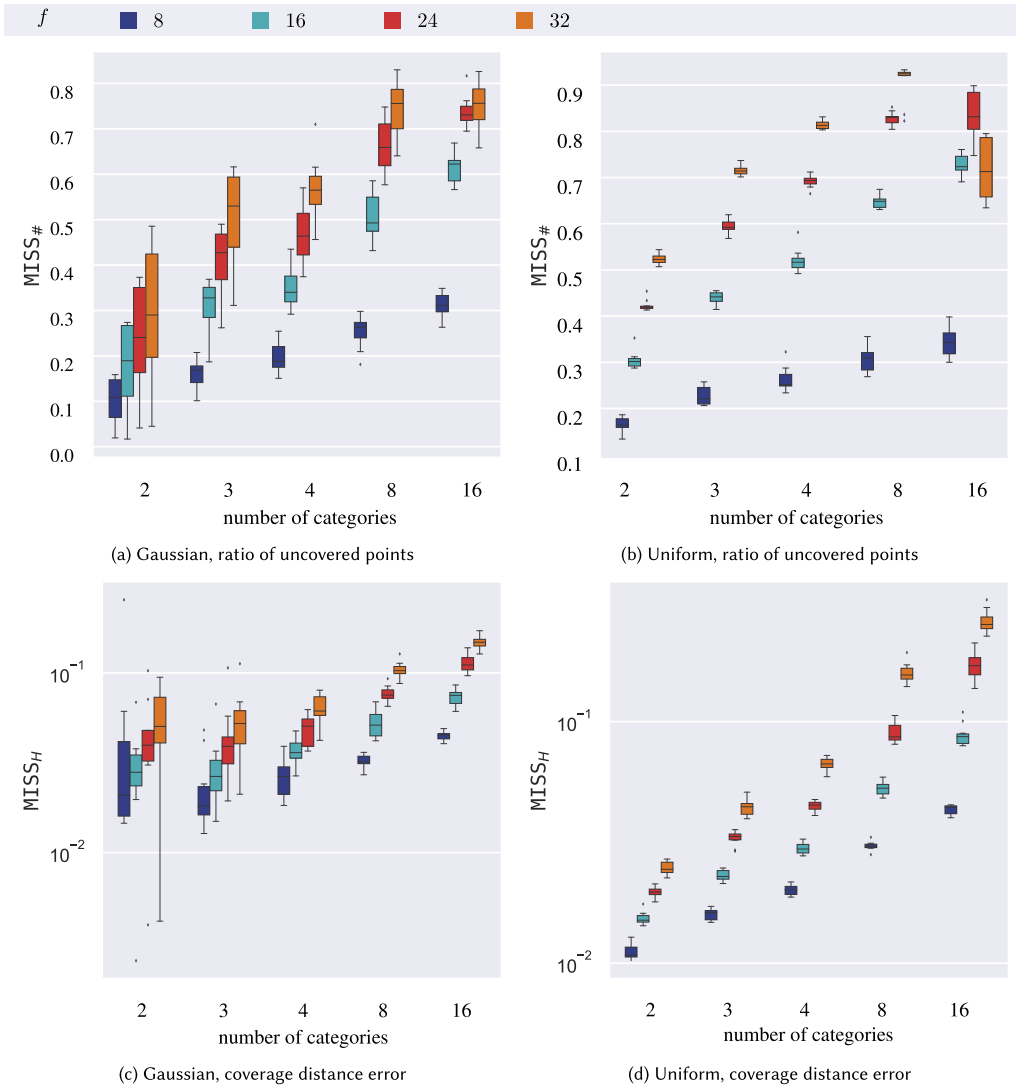


Fig. 18. Box plot of coverage error with respect to number of categories of Gaussian (left) and Uniform instances (right). Each box shows the results of the greedy heuristic with specific minimum font size, on 10 instances. The coverage distance error of each point is normalized by dividing by the diagonal length of its corresponding cover rectangle.

aim at getting labels that represent the underlying data quantitatively by scaling labels according to the number of points they represent, it is desirable to get solutions of low density variance. We do not see a direct way to set constraints for candidate rectangles such that the selected cover rectangles have similar density. However, the purpose of this experiment is to explore the relation of the solution quality density, with our designed parameters.

For the constraints dealing with aspect ratio (Figure 19(a), Figure 19(b)) and misrepresentation tolerance (Figure 19(c) and (d)) we do not get any clear pattern, no matter in comparison of different number of categories or different parameter settings. However, the constraint

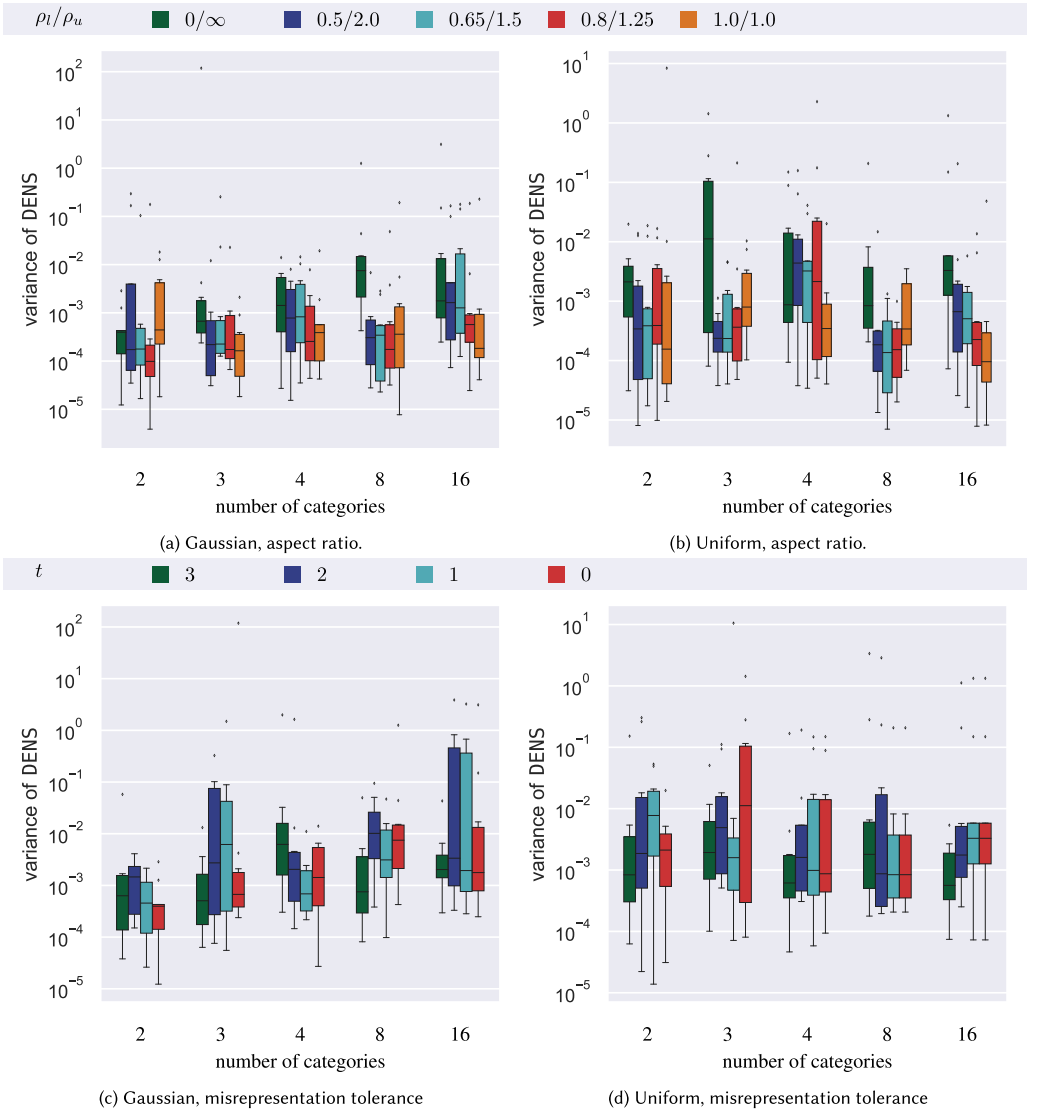
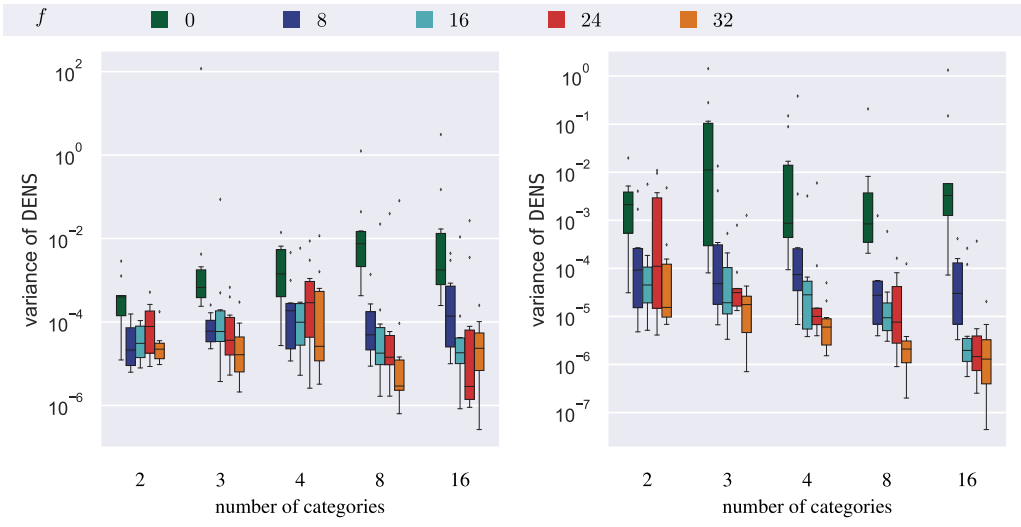


Fig. 19. Box plot of density and aspect ratio/misrepresentation tolerance, grouped in number of categories. Each box shows the results with a specific parameter, in perspective of ratio/misrepresentation tolerance, on 10 instances.

minimum font size (Figure 20(a) and (b)) affects the density variances in a more clear pattern: the larger the minimum font size, the smaller the density variance. One possible reason could be that with the increase of the minimum font size, the area range of cover rectangles get smaller, thus the density variance decreases; see an example in Figure 21.

5.4 Case Study – U.S. Crop Harvest

We use our greedy heuristic to produce Worbel visualizations for the US-Crops data set, to see the effectiveness of the visualizations on real-world data. These pictures can easily be produced



(a) Gaussian, font (b) Uniform, font.

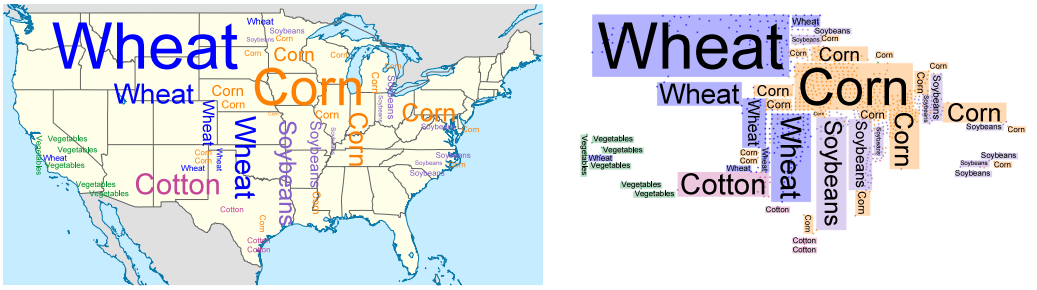
Fig. 20. Box plot of density and minimum font size, grouped in number of categories. Each box shows the results with a specific minimum font size, on 10 instances.

(a) Without constraints (b) With minimum font size 32

Fig. 21. Density variance. Examples of different data sets with high density variance (a) and low density variance (b).

on a less powerful machine (see Section 5.1). We used the following parameters to compute the visualizations in this case study (unless mentioned otherwise), $\rho_l = 0.75$, $\rho_u = 2$, $t = 2$, and $f = 16$.

We start by considering the full data set, US-Crops, comprising 3076 points. The result is shown in Figure 1 and was computed in 98 seconds. If we look at the Worbel on the map of the U.S. (Figure 1(a)), we see a fairly dense set of labels that leaves few open spaces. Around the sides of the map, we see that labels can extend outwards. The algorithm clusters a few points or assigns labels to singletons/outliers in these cases, resulting a labeling that resembles a classic point feature



(a) (b)

Fig. 22. Worbels of the US-Crops-filtered data set, using parameters $\rho_l = 0.75$, $\rho_u = 2$, $t = 2$, and $f = 16$, and produced in 15 seconds. (a) Drawn on a map. (b) Drawn over the data points; rectangles outline the computed area per label.

labeling. Along the east coast, we see many such labels. However, in the middle of the map, many points are clustered and covered by single label. These labels resemble an area labeling of the map, indicating that the areas covered by labels predominantly have points of that same category. For example, this happens with the labels Wheat and Corn, around Montana/North Dakota, and Iowa/Wisconsin/Illinois, respectively. Overall, solving the RPFA problem here seems to result in a labeling that is a nice hybrid between area labeling and (single) point labeling.

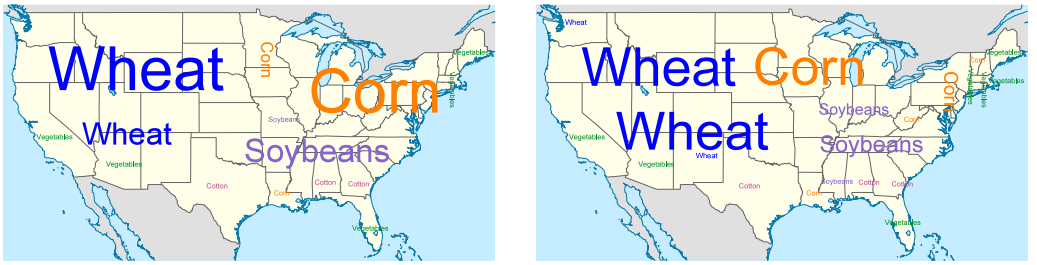
If we look at the computed rectangles and the uncovered data points (Figure 1(b)), we see that most rectangles stack nicely and cover the represented points as intended. The minimum font size prevents some points from being covered; either their label would intersect some other label in the Worbel (see, e.g., the isolated points in the west of the U.S.), or their label would intersect too many points with a different label (such as in Alabama/Georgia area and Oklahoma).

Our findings for US-Crops-filtered are similar. Figure 22 shows the Worbel for this data set, computed in about 15 seconds. Since there are less points, and fewer outliers in this data set, we see larger labels throughout this Worbel. The open areas did not contain any points, because of the filtering, and the more densely packed areas leave some points uncovered, again due to the minimum font size. Similar as before, areas with many points result in tightly packed area labelings, while for sparser areas a single point labeling is produced. Also note that, while the parameters for aspect ratio are not very strict, we still produce rectangles close to the aspect ratio of the respective labels.

In Figure 23 we show a Worbel for the US-Crops-State data set, produced in less than a second. This data set contains a single data point per state, and hence is very sparse. In Figure 23(a) we see the Worbel generated with the input parameters as before. However, since there are very few points, we decided to tune down the tolerance for misrepresentation to zero, since the error we make is significantly higher than for the larger data sets. Setting $t = 0$ produces the Worbel in Figure 23(b). Even though there are only 48 points, we get a clustering of Wheat, Corn, and Soybeans in the central states. Along the boundary of the map, states are labeled as singletons, so the produced Worbel is still a nice hybrid between area labeling and point labeling, even for sparse data.

6 DISCUSSION AND CONCLUDING REMARKS

The Worbel hybrid visualization scheme proposed in this article is designed to provide an aggregated textual labeling for geospatial data points. Our contributions include a formalization of the problem underlying the aggregation of geospatial data via the RPFA problem, and a proof establishing that RPFA is NP-complete.



(a) Worbel with tolerance for misrepresentation

(b) Worbel without misrepresentation

Fig. 23. Worbels of the US-Crops-state data set, overlaid on a map, and produced in <1 second. (a) Using parameters $\rho_l = 0.75$, $\rho_u = 2$, $t = 2$, and $f = 16$. (b) Using parameters $\rho_l = 0.75$, $\rho_u = 2$, $t = 0$, and $f = 16$.

In view of this algorithmic lower bound, we propose and implement two algorithms for computing Worbels. The first is designed to find theoretically optimal Worbels for small to medium-size instances of RPFA, and is based on a MAXSAT model (which turned out to be significantly more scalable than ILP and CSP-based approaches in our initial experiments). We complemented this with an iterative heuristic algorithm. Our experiments showed that the running time performance of this heuristic outscales the exact MAXSAT approach by several orders of magnitude and can produce Worbels for instances with thousands of data points in a matter of seconds, even on standard work stations. At the same time, for the vast majority of tested instances the Worbels produced by the heuristic only exhibit an overhead of at most 10% additional aggregate labels.

The experimental results also give rise to the practically interesting option of combining the strengths of both approaches. It turned out that the size of the candidate set of rectangles is a much better predictor of the computation time than the size of the point set or the number of different point categories alone. Since both methods begin by computing these candidate sets, we can compute them and then depending on the resulting size either invoke the MAXSAT solver (if we expect fast termination) or invoke the greedy heuristic (if the candidate set size exceeds an adjustable threshold). Further research should investigate more sophisticated methods to prune the candidate set by discarding rectangles, which are identified as suboptimal without compromising the overall solution quality, and thus accelerate computation times for both algorithms.

From the experiments, we can conclude that the parameters of the RPFA problem behave as expected, e.g., increasing minimum font size results in fewer labels. The parameters did not show any predictable behavior on the density metric, but we can still use it to find individual solutions with the property that rectangle area is proportional to number of underlying data points. These instances hence portrait an important characteristic of traditional word clouds.

An interesting observation from our case study on the US crop data is that the resulting Worbels actually combine features of area and point labeling. In homogeneous groups of points with the same label, a single large rectangle can be placed, effectively labeling the entire covered area. In areas with more heterogeneous point features, labels for smaller clusters or even singleton points are placed, much as in traditional point feature label placement. As a result, Worbels implicitly decrease the information density of text-based data visualizations on maps in areas of low entropy by placing fewer labels, while showing more detailed and less aggregated information in areas of higher entropy by placing more but smaller labels. An important open question is whether these promising properties of Worbels translate into quantifiable usability and readability advantages when compared to maps with individual point labels and/or less precise geo word clouds or tag maps.

Ultimately, we believe that integrating Worbels into interactive visual analytics tools or GIS systems provides an added value for the users of these systems. One opportunity in this context is that analysts can modify and improve an initially computed Worbel by locally fine-tuning quality parameters or adding context-dependent constraints and thus creating Worbels according to their individual preferences in a human-in-the-loop process. This will require more advanced algorithms for computing visually stable Worbels on a dynamic input.

REFERENCES

[1] Anna Adamaszek and Andreas Wiese. 2013. Approximation schemes for maximum weight independent set of rectangles. In *Proceedings of the 54th Symposium on Foundations of Computer Science*. IEEE Computer Society, 400–409. DOI: https://doi.org/10.1109/FOCS.2013.50

[2] Pankaj K. Agarwal and Jiangwei Pan. 2020. Near-linear algorithms for geometric hitting sets and set covers. *Discrete and Computational Geometry* 63, 2 (2020), 460–482. DOI: https://doi.org/10.1007/s00454-019-00099-6

[3] Pankaj K. Agarwal, Marc J. van Kreveld, and Subhash Suri. 1998. Label placement by maximum independent set in rectangles. *Computational Geometry: Theory and Applications* 11, 3–4 (1998), 209–218. DOI: https://doi.org/10.1016/S0925-7721(98)00028-5

[4] Hee-Kap Ahn, Sang Won Bae, Erik D. Demaine, Martin L. Demaine, Sang-Sub Kim, Matias Korman, Iris Reinbacher, and Wanbin Son. 2011. Covering points by disjoint boxes with outliers. *Computational Geometry: Theory and Applications* 44, 3 (2011), 178–190. DOI: https://doi.org/10.1016/j.comgeo.2010.10.002

[5] Pradeesha Ashok, Sudeshna Kolay, Neeldhara Misra, and Saket Saurabh. 2015. Unique covering problems with geometric sets. In *Proceedings of the 21st Computing and Combinatorics Conference*. Springer, 548–558. DOI: https://doi.org/10.1007/978-3-319-21398-9_43

[6] Pradeesha Ashok, Sudeshna Kolay, and Saket Saurabh. 2017. Multivariate complexity analysis of geometric red blue set cover. *Algorithmica* 79, 3 (2017), 667–697. DOI: https://doi.org/10.1007/s00453-016-0216-x

[7] Mathieu Barrault. 2001. A methodology for placement and evaluation of area map labels. *Computers, Environment and Urban Systems* 25, 1 (2001), 33–52. DOI: https://doi.org/10.1016/S0198-9715(00)00039-9

[8] Lukas Barth, Sara Irina Fabrikant, Stephen G. Kobourov, Anna Lubiw, Martin Nöllenburg, Yoshio Okamoto, Sergey Pupyrev, Claudio Squarcella, Torsten Ueckerdt, and Alexander Wolff. 2014. Semantic word cloud representations: Hardness and approximation algorithms. In *Proceedings of the 11th Latin American Symposium on Theoretical Informatics*. Springer, 514–525. DOI: https://doi.org/10.1007/978-3-642-54423-1_45

[9] Sergey Bereg, Sergio Cabello, José Miguel Díaz-Báñez, Pablo Pérez-Lantero, Carlos Seara, and Inmaculada Ventura. 2012. The class cover problem with boxes. *Computational Geometry: Theory and Applications* 45, 7 (2012), 294–304. DOI: https://doi.org/10.1016/j.comgeo.2012.01.014

[10] Sujoy Bhore, Guangping Li, and Martin Nöllenburg. 2020. An algorithmic study of fully dynamic independent sets for map labeling. In *Proceedings of the 28th European Symposium on Algorithms*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 19:1–19:24. DOI: https://doi.org/10.4230/LIPIcs.ESA.2020.19

[11] Kevin Buchin, Daan Creemers, Andrea Lazzarotto, Bettina Speckmann, and Jules Wulms. 2016. Geo word clouds. In *Proceedings of the 9th IEEE Pacific Visualization Symposium*. IEEE Computer Society, 144–151. DOI: https://doi.org/10.1109/PACIFICVIS.2016.7465262

[12] Robert D. Carr, Srinivas Doddi, Goran Konjevod, and Madhav V. Marathe. 2000. On the red-blue set cover problem. In *Proceedings of the 11th Symposium on Discrete Algorithms*. SIAM, 345–353.

[13] Parinya Chalermsook and Bartosz Walczak. 2021. Coloring and maximum weight independent set of rectangles. In *Proceedings of the 32nd Symposium on Discrete Algorithms*. SIAM, 860–868. DOI: https://doi.org/10.1137/1.9781611976465.54

[14] Timothy M. Chan and Nan Hu. 2015. Geometric red-blue set cover for unit squares and related problems. *Computational Geometry: Theory and Applications* 48, 5 (2015), 380–385. DOI: https://doi.org/10.1016/j.comgeo.2014.12.005

[15] Ming-Te Chi, Shih-Syun Lin, Shiang-Yi Chen, Chao-Hung Lin, and Tong-Yee Lee. 2015. Morphable word clouds for time-varying text data visualization. *IEEE Transactions on Visualization and Computer Graphics* 21, 12 (2015), 1415–1426. DOI: https://doi.org/10.1109/TVCG.2015.2440241

[16] Kenneth L. Clarkson and Kasturi R. Varadarajan. 2007. Improved approximation algorithms for geometric set cover. *Discrete and Computational Geometry* 37, 1 (2007), 43–58. DOI: https://doi.org/10.1007/s00454-006-1273-8

[17] Jessica Davies and Fahiem Bacchus. 2014. MaxHS. Retrieved May 28, 2021 from http://maxhs.org/.

[18] Erik D. Demaine, Uriel Feige, MohammadTaghi Hajiaghayi, and Mohammad R. Salavatipour. 2008. Combination can be hard: Approximability of the unique coverage problem. *SIAM journal on Computing* 38, 4 (2008), 1464–1483. DOI: https://doi.org/10.1137/060656048

[19] Borden D. Dent, Jeffrey S Torguson, and Thomas W. Hodler. 2009. *Cartography: Thematic Map Design* (6th ed.). McGraw-Hill.
[20] Michael Formann and Frank Wagner. 1991. A packing problem with applications to lettering of maps. In *Proceedings of the 7th Symposium on Computational Geometry*. 281–288. DOI: https://doi.org/10.1145/109648.109680
[21] M. R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.
[22] Jan-Henrik Haunert and Alexander Wolff. 2017. Beyond maximum independent set: An extended integer programming formulation for point labeling. *International Journal of Geo-Information* 6, 11 (2017), 342:1–342:20. DOI: https://doi.org/10.3390/ijgi6110342
[23] Fabian Klute, Guangping Li, Raphael Löffler, Martin Nöllenburg, and Manuela Schmidt. 2019. Exploring semi-automatic map labeling. In *Proceedings of the 27th Advances in Geographic Information Systems*. ACM, 13–22. DOI: https://doi.org/10.1145/3347146.3359359
[24] Chenlu Li, Xiaoju Dong, and Xiaoru Yuan. 2018. Metro-Wordle: An Interactive Visualization for Urban Text Distributions Based on Wordle. *Vis. Informatics* 2, 1 (2018), 50–59. DOI: https://doi.org/10.1016/j.visinf.2018.04.006
[25] Alan M. MacEachren, Anuj R. Jaiswal, Anthony C. Robinson, Scott Pezanowski, Alexander Savelyev, Prasenjit Mitra, Xiao Zhang, and Justine I. Blanford. 2011. SensePlace2: GeoTwitter analytics support for situational awareness. In *Proceedings of the Visual Analytics Science and Technology*. IEEE, 181–190.
[26] Joe Marks and Stuart Shieber. 1991. *The Computational Complexity of Cartographic Label Placement*. Technical Report. Harvard University. Retrieved 10 March 2023 from http://www.eecs.harvard.edu/~shieber/Biblio/Papers/label.pdf.
[27] Maurizio Patrignani. 2013. Planarity testing and embedding. In *Proceedings of the Handbook on Graph Drawing and Visualization*. Roberto Tamassia (Ed.), Chapman and Hall/CRC, 1–42.
[28] Martin Reckziegel, Muhammad Faisal Cheema, Gerik Scheuermann, and Stefan Jänicke. 2018. Predominance tag maps. *IEEE Transactions on Visualization and Computer Graphics* 24, 6 (2018), 1893–1904. DOI: https://doi.org/10.1109/TVCG.2018.2816208
[29] Martin Reckziegel and Stefan Jänicke. 2019. Time varying predominance tag maps. In *Proceedings of the 30th Visualization Conference*. IEEE, 231–235. DOI: https://doi.org/10.1109/VISUAL.2019.8933654
[30] Maxim A. Rylov and Andreas W. Reimer. 2014. A comprehensive multi-criteria model for high cartographic quality point-feature label placement. *Cartographica* 49, 1 (2014), 52–68. DOI: https://doi.org/10.3138/carto.49.1.2137
[31] Aidan Slingsby, Jason Dykes, Jo Wood, and Keith C. Clarke. 2007. Interactive tag maps and tag clouds for the multiscale exploration of large spatio-temporal datasets. In *Proceedings of the 11th Conference on Information Visualisation*. IEEE Computer Society, 497–504. DOI: https://doi.org/10.1109/IV.2007.71
[32] Tycho Strijk. 2001. *Geometric Algorithms for Cartographic Label Placement*. Ph. D. Dissertation. Universiteit Utrecht.
[33] Dennis Thom, Harald Bosch, Steffen Koch, Michael Wörner, and Thomas Ertl. 2012. Spatiotemporal anomaly detection through visual analysis of geolocated Twitter messages. In *Proceedings of the Pacific Visualization*. IEEE, 41–48.
[34] Leslie G. Valiant. 1981. Universality considerations in VLSI circuits. *IEEE Transactions on Computers* 100, 2 (1981), 135–140. DOI: https://doi.org/10.1109/TC.1981.6312176
[35] Marc J. van Kreveld. 2004. Geographic information systems. In *Proceedings of the Handbook of Discrete and Computational Geometry, Second Edition*. Chapman and Hall/CRC, 1293–1314. DOI: https://doi.org/10.1201/9781420035315.ch58
[36] Marc J. van Kreveld, Tycho Strijk, and Alexander Wolff. 1999. Point labeling with sliding labels. *Computational Geometry: Theory and Applications* 13, 1 (1999), 21–47. DOI: https://doi.org/10.1016/S0925-7721(99)00005-X
[37] Fernanda B. Viégas, Martin Wattenberg, and Jonathan Feinberg. 2009. Participatory visualization with wordle. *IEEE Transactions on Visualization and Computer Graphics* 15, 6 (2009), 1137–1144. DOI: https://doi.org/10.1109/TVCG.2009.171
[38] Pinhas Yoeli. 1972. The logic of automated map lettering. *The Cartographic journal* 9, 2 (1972), 99–108. DOI: https://doi.org/10.1179/000870472787352505

Received 29 July 2022; revised 10 March 2023; accepted 16 May 2023