Optimal straight-line labels for island groups

Optimal Straight-line Labels for Island Groups

Arthur van Goethem[*] Marc van Kreveld[†] Andreas Reimer[‡] Maxim Rylov[‡] Bettina Speckmann[*]

1 Introduction

Maps are used to solve a wide variety of tasks, ranging from navigation to analysis. Often, the quality of a map is directly related to the quality of its labelling. Consequently, a lot of research has focussed on the automatization of the labelling process (for an overview see [11]). Surprisingly the (automated) labelling of island groups has received little attention so far. This is at least partially caused by the lack of cartographic principles. Though extensive guidelines for map labelling exist (e.g., [6, 12]), information on the labelling of groups of islands is surprisingly sparse.

We take an algorithmic approach and focus on optimal solutions to the problem using different settings. Comparing manual and optimal labels may give insight in subconscious rules applied by cartographers.

Contribution. We define a formal framework for island labelling. The framework spawns a large series of unexplored computational geometry problems. In this paper we start by looking at a straight-line label. In Section 2 we describe two algorithms for a straight-line label that is, or is not, allowed overlap with islands. Section 3 discusses several extensions to these algorithms for closely related problems.

Framework. We assume the input to the island labeling problem is a set S of k islands, given as simple polygons P_1, ..., P_k, with n vertices in total. We abstract away from the label itself and assume the label can be represented by its placement area. To this end we assume the label has a baseline that is either straight, circular, or a Bézier curve. The placement area is defined by perpendicularly extruding the baseline by a (possibly zero) height. Besides the shape, labels may be optimal with respect to three characteristics. Firstly, the distance to the islands may be computed using different points of measure. Specifically, these include the centroid of the polygon, the point on the polygon closest to the label, and the complete area of the polygon. Secondly, different distance measures may be used. Three distance measures used in cartography are included, minimizing: the maximum distance (*min-max*), the sum of distances (*min-sum*), the sum of squared distances (*min-sum-sq*). Lastly, we (dis)allow overlap of the label with the islands. For all labels we make the assumption that the label is long enough to measure distance perpendicularly.

In this paper we focus on straight-line labels optimizing the min-max distance to the closest point of each island. We are interested in strictly optimal labels to prevent subjectivity. For practical labelling purposes a trivial discretization may be sufficient.

Related work. Many algorithms have been developed for labelling maps [11]. Labelling islands groups has received less attention, but was addressed by Van Kreveld and Slechter [7]. Their algorithm places a non-overlapping label in the position minimizing the maximum distance from the label to each island of the group. They, however, require labels to be horizontal and discretize the space of possible label placements.

Several known algorithms can directly be applied to labelling. All algorithms assume that we are dealing with a point set, which is true if we have a fixed point of measurement for each island. Furthermore, all algorithms apply to a straight-line label that may overlap islands. The rotating calipers algorithm [9] optimizes the min-max distance in $O(n \log n)$ time. Using known results from Dey [3], Brodal and Jacob [2] and Edelsbrunner and Welzl [4], the label optimizing the min-sum distance can be computed in $O(n + k^{4/3} \log k)$ time. Finally, Deming regression optimizes the min-sum-sq distance in $O(n)$ time. For circular arcs the minimum width annulus [1] solves the min-max distance problem in $O(n^2)$ time.

2 Optimal, straight-line label

To compute an optimal, straight-line label we make use of the arrangement of possible label positions in *dual space* [1]. We start with the simple case consisting of a straight-line label that is allowed overlap with the islands. We focus on the min-max distance measure to the closest point of each island. An island intersecting the label has distance 0. For now we assume labels have no height and, hence, are represented by a single line. As stated before, we assume we have a set S of k islands, given as simple polygons P_1, ..., P_k, with n vertices in total. We also assume that all labels are sufficiently long to measure all distances perpendicular.

[*]Dept. of Mathem. and Computer Science, TU Eindhoven, The Netherlands, [a.i.v.goethem|b.speckmann]@tue.nl.
Supported by the Netherlands Organisation for Scientific Research (NWO) under project no. 612.001.102 (AvG) and no. 639.023.208 (BS).

[†]Dept. of Information and Computer Sciences, Utrecht University, The Netherlands, m.j.vankreveld@uu.nl

[‡]Institute of Geography, Heidelberg University, Germany, [andreas.reimer|maxim.rylov]@geog.uni-heidelberg.de

Overlapping label. As we are only interested in the distance to the closest point, we first compute the convex hull of each island in $O(n)$ time [8] and replace each island by its convex hull. The k convex polygons in primal space, become k funnels in dual space. The *top and bottom boundaries* of these funnels are x-monotone polylines. Between each pair of islands there exist at most four tangents. Consequently, the top and bottom of any pair of funnels intersect each other at most four times. The boundaries of the funnels form a set of $2k$ x-monotone, 2-intersecting polylines with $O(n)$ vertices total.

For any fixed rotation, the furthest vertex (or edge) below the label in primal space, is on the upper envelope of the lower boundaries in dual space. As all the funnel-boundaries are 2-intersecting, we can compute the upper envelope of the lower boundaries in $O(n + k \log k)$ time and it has complexity $O(n)$. A similar argument holds for the furthest vertex above and the lower envelope of the upper boundaries.

Lemma 1 *The optimal solution for a fixed rotation is halfway between the upper- and lower-envelope.*

The optimal solutions for each rotation are located on a polygonal line that is exactly centered between the upper- and lower- envelope, and it has $O(n)$ complexity. When an optimal solution intersect all islands it is not required to have equal distances to either side. The solution having equal distances to the closest vertex (edge) on either side, however, is also a valid optimal solution.

Lemma 2 *For each segment s of the solution-line we can compute the optimal position in $O(1)$ time.*

Proof. Let x_{start} be the x-coordinate of the start of s. Let d_{start} be the vertical distance to the upper- (or lower-) envelope at x_{start}. While we move along s the distance to the upper envelope changes by a linear factor c_1 in x. For a shift of δ along the x-axis, the vertical distance in dual space is $f_s(\delta) = d_{start} + c_1 * \delta$. The distance to the closest point in primal space is

$$g_s(\delta) = \frac{(d_{start} + c_1 * \delta)^2}{(x_{start} + \delta)^2 + 1} \qquad (1)$$

The optimal position is at the minimum over the domain given by segment s. This minimum can be at an endpoint of s or in the middle. □

Theorem 3 *We can compute the optimal straight-line label minimizing the maximum minimum distance over all islands in $O(n + k \log k)$ time.*

Intersection-free label. Positions exactly halfway between the upper- and lower-envelope may result in labels that overlap one or more islands. For intersection-free labels we require more information. An optimal intersection-free label need not necessarily have an equal distance to the furthest closest point on either side. Therefore, we compute the complete arrangement of all $2k$ boundaries. In this arrangement we find the optimal intersection-free solution in $O(nk + k^2 \log k)$ time.

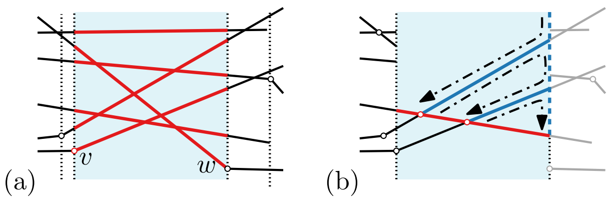

Figure 1: (a) The section of the arrangement (red) between vertices v and w, with $C = 7$ intersections. (b) The number of edges traversed to insert a new line with c intersections is $O(c)$.

Lemma 4 *The arrangement formed by all $2k$ boundaries can be computed in $O(nk)$ time.*

Proof. Make a single sorted list V of all vertices (excluding intersections), sorted along the x-axis, in $O(n \log k)$ time by merging the vertex-lists of the $2k$ boundaries. Sort all boundaries by decreasing slope of their first segment in $O(k \log k)$ time.

We compute the *complete arrangement* by performing a sweep along the x-axis. Define a *section* (v, w) of the complete arrangement as the arrangement consisting of all vertices u with $v_x \leq u_x < w_x$ and all maximal segments of edges having endpoints s, t, such that $v_x \leq s_x < w_x$ and $v_x \leq t_x < w_x$ (see Fig. 1(a)).

Let v_i, w_{i+1} be two consecutive vertices in V having index i and $i+1$ respectively. In step i of the sweep we compute section (v_i, w_{i+1}) of the complete arrangement. The start and end of the arrangement are included by adding dummy vertices to V at minus infinity and infinity. Each section is computed in $O(k+C)$ time, where C is the number of intersections in this section.

As each step is between consecutive vertices in V, within each section all boundaries form straight lines. Within a section (v, w) we introduce all boundary-lines in sorted order (being sorted bottom to top at v_x). This order is known at the start of the algorithm and maintained during the sweep.

When introducing a line we find all intersections with the previously introduced lines. To find the intersections we move in a clockwise fashion through the arrangement built so far (see Fig. 1(b)). Any traversed line is either a boundary introduced earlier (starting lower) and ending higher, or the "virtual line" at w_x. Hence, the number of distinct lines we traverse is at most $O(c)$, where c is the number of intersections we detect by introducing this line. As lines are 1-intersecting, the traversed edges in the worst case form a Davenport-Schinzel-sequence [10] of order 2 using c symbols, having complexity $O(c)$. Thus, when introducing a single boundary-line we spend at most $O(c)$ time.

The total number of intersections in a section is C. We may spend an extra constant amount of work per boundary, so introducing all lines takes $O(k + C)$ time. While introducing the lines we use insertion sort to obtain the sorted order at the end of each section in $O(k + C)$ time as well.

Because the number of intersections in the complete arrangement is bounded by $O(k^2)$, the algorithm runs in $O(nk + k^2) = O(nk)$ time. □

During creation of the arrangement we also keep track of the left-most point of each face. This point is uniquely defined as all lines are strictly x-monotone. The non-closed faces on the left of the arrangement maintain pointers to both infinite rays. We also store for each face if it is covered by a funnel. Faces covered by a funnel are defined to be *illegal*. An edge of the arrangement separating two illegal faces is also considered *illegal*, all other edges are *legal*.

Lemma 5 *The centerline may cross the arrangement $O(nk)$ times.*

Lemma 6 *The centerline can be inserted in the arrangement in $O(nk)$ time.*

If, for a fixed rotation, the point on the centerline is illegal, then clearly the closest legal point with the same rotation is optimal. The closest legal points form subedges on the edges of the arrangement (see Fig. 2).

Lemma 7 *We can find the closest legal subedges above the centerline in $O(nk + k^2 \log k)$ time.*

Proof. Remove from the computed arrangement all edges that are illegal in $O(n + k^2)$ time. Let a *chain* be a maximal series of consecutive edges connected by degree two vertices. The resulting arrangement has, excluding the centerline, $O(k^2)$ non-intersecting chains with $O(n+k^2)$ vertices total. We merge all vertices of the chains into a sorted list in $O((n+k^2) \log k)$ time. Now we merge the sorted list with the sorted vertices of the centerline in $O(nk)$ time to get a sorted list of all possible events.

We create a red-black tree (RB-tree) of all chains starting at the left-most end of the arrangement sorted by vertical order in $O(k \log k)$ time. The centerline is also added to the RB-tree. To all leafs of the RB-tree we add pointers to the neighboring leafs so that we can query for neighbors in $O(1)$ time.

We now move a sweepline over all vertices using the sorted list. Two types of events may occur. Firstly, when the first vertex of a new chain is reached, we add the chain to the red-black tree in $O(\log k)$ time. Similarly, when the last vertex of a chain is reached, we remove it. Secondly, when a chain intersects the centerline, we switch their positions in the red-black tree in $O(1)$ time.

At each event we can in $O(1)$ time determine the closest legal chain above the centerline. This may possibly be the centerline itself if it currently legal. Hence, we can trace the subedges above the centerline in $O(nk + k^2 \log k)$ time. □

The closest point to the centerline may be on the closest legal subedge above or below the centerline or may be on the centerline itself. However, the number of times where the point switches between these lines is bounded by $O(nk)$.

Lemma 8 *The closest legal line can switch at most $O(nk)$ times between the lower and upper legal line.*

Proof. Let W be the ordered set of vertices of the complete arrangement (including centerline) by x-coordinate. Let v and w be two consecutive vertices in W and $v_x \neq w_x$. We define a *slab* as the set of maximal segments of edges having endpoints s, t with $s_x = v_x$ and $t_x = w_x$ (see Fig. 3). There are $O(nk)$ vertices in the arrangement, so also $O(nk)$ slabs. For each slab all the segments are straight and non-intersecting. The closest legal point can switch at most once per slab and, thus, $O(nk)$ total. □

We can easily find the points on the centerline where the closest legal line changes and insert an extra vertex on the centerline. For each new edge of the centerline, the distance to the furthest closest point

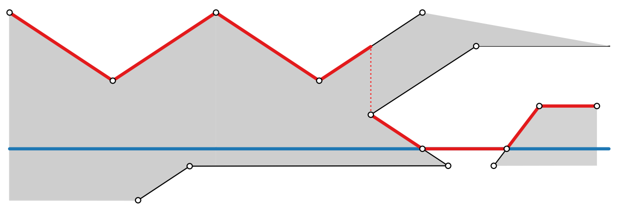

Figure 2: Centerline (blue), illegal faces (grey) and the closest legal subedges (red).

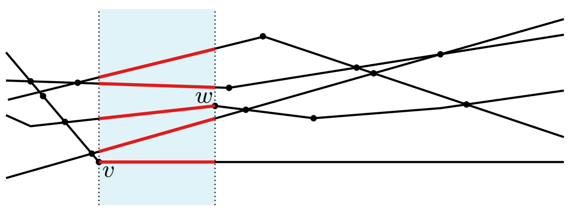

Figure 3: A *slab* of the complete arrangement (blue) and the belonging segments of edges (red). There are $O(nk)$ slabs in the complete arrangement.

changes linearly by c_1 and the distance to the closest legal point also changes linearly by c_2. Hence, similar to the overlapping version, we can create a closed formula for each segment s based on the shift δ along the x-axis.

$$g_s(\delta) = \frac{(d_{start} + o_{start} + (c_1 + c_2) * \delta)^2}{(x_{start} + \delta)^2 + 1} \quad (2)$$

Theorem 9 *We can compute the optimal straight-line label optimizing the min-max distance that is non-overlapping in $O(nk + k^2 \log k)$ time.*

3 Extensions

In the previous section we have given algorithms for a (non-)overlapping label using the min-max distance. The basic idea can also be applied to solve different problems. In this section we give two examples.

Minimizing the sum of distances. It is easy to show that an optimal solution has equally many islands placed to either side of the label (and may possibly intersect some others). Hence, for a fixed rotation the optimal (possibly overlapping) solution can easily be found in $O(n + k \log k)$ time. For an intersection-free label in a fixed rotation, the optimal placement is the legal placement closest to the above solution.

In both cases it holds that if an optimal solution exists not tangent to any island, then there is also an optimal solution that is. Thus, the optimal solution in dual space is located on the edges of the arrangement.

We can compute the arrangement in $O((n + k^2) * \log k)$ time using a sweep line. While sweeping the arrangement we keep track of the faces having an equal number of islands above and below it. We compute the summed distance at the start and its linear change in the first slab in $O(k)$ time. For each event we update the linear change and the summed distance at the start in $O(1)$ time. We can compute the optimal position per edge in $O(1)$ time. Finding the closest legal edge can be done in $O(\log k)$ time as well. Hence, in $O((n+k^2) \log k)$ time we can find the optimal non-overlapping label.

Non-zero-height labels. Zero-height labels are not useful for real labelling. For an overlapping label, the optimal non-zero-height label is always centered on the optimal zero-height solution.

If we wish to place an intersection-free label of height h, this is equivalent to offsetting all islands by $h/2$ and computing a zero-height label. The vertices of the islands in primal space, however, become circular arcs. Consequently, in dual space we do not have polylines of line-segments, but of curves. A vertex $p = (p_x, p_y)$ of the top side of an islands being offset by a distance $h/2$ becomes a curve $p^* = p_x * x - p_y - \sqrt{x^2 + 1} * h/2$ in dual space. Vertices on the bottom side of islands become curves $p^* = p_x * x - p_y + \sqrt{x^2 + 1} * h/2$.

We can still compute the arrangement, the centerline, and the closest legal segments in $O(nk+k^2 \log k)$ time. For each segment we can in $O(1)$ time compute the optimal position, so we can find the global optimum in $O(nk+k^2 \log k)$ time. A similar argument can be made for the min-sum distance measure resulting in an $O((n + k^2) \log k)$ time algorithm.

Future work. In this paper we gave an $O(nk)$ algorithm to compute the arrangement. Technically, this is not necessary as a sweep-line algorithm can obtain the same $O(nk + k^2 \log k)$ time-bounds for the complete algorithm. However, the entire first section of our algorithm runs in $O(nk)$ time. In the worst case the centerline may intersect the arrangement at most $O(nk)$. This bound is also achievable. The final extra log-factor in $O(k^2 \log k)$ only turns up once. In future work we might investigate if this log-factor can also be removed, reaching the $O(nk)$ runtime. Current investigations, however, indicate a possible relation to the Sorting-(X+Y) problem [5].

References

[1] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry, 3rd edition.* Springer, 2008.

[2] G. Brodal and R. Jacob. Dynamic planar convex hull. In *Proc. 43rd Annual IEEE Symp. on Foundations of Computer Science*, pages 617–626. ACM, 2002.

[3] T. Dey. Improved Bounds for Planar k-Sets and Related Problems. *Discrete Comput. Geom.*, 19(3):373–382, 1998.

[4] H. Edelsbrunner and E. Welzl. Constructing belts in two-dimensional arrange-ments with applications. *SIAM J. on Computing*, 15(1):271–284, 1986.

[5] M. Fredman. How good is the information theory bound in sorting? *Theoret. Comput. Sci.*, 1:355–361, 1976.

[6] E. Imhof. Positioning names on maps. *American Cartographer*, 2(2):128–144, 1975.

[7] M. van Kreveld and T. Slechter. Automated label placement for groups of islands. *Proc. 22th Int. Cart. Conf.*, 2005.

[8] A. Melkman. On-line construction of the convex hull of a simple polyline. *Information Processing Letters*, 25(1):11–12, 1987.

[9] M. Shamos. Computational geometry, 1978. Ph.D. thesis. Dept. of CS, Yale Univ.

[10] C. Toth, J. O'Rourke, and J. Goodman. *Handbook of Discrete and Computational Geometry.* CRC press, 2004.

[11] A. Wolff and T. Strijk. A map labeling bibliography, 2009. http://i11www.iti.uni-karlsruhe.de/ awolff/map-labeling/bibliography/.

[12] C. Wood. Descriptive and Illustrated Guide for Type Placement on Small Scale Maps. *Carto. J.*, 37(1):5–18, 2000.