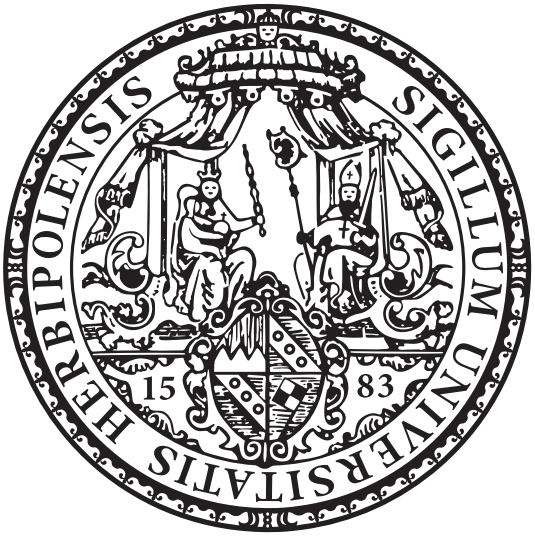
Master Thesis

Boundary Labeling with Polygon Obstacles

Annika Förster

Date of Submission:	30. October 2021
Advisors:	Prof. Dr. Alexander Wolff
	Jun.-Prof. Dr. Thomas van Dijk



Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik I
Algorithmen und Komplexität

Abstract

Labels are key to understanding not only geographical maps, but also technical drawings or diagrams. They can be placed directly on the image or on a boundary box around it. In that case the map features are connected with the labels by so-called *leaders* which can be straight lines, curves or polylines. In this thesis we label the edges of a polygon. This polygon is an *obstacle*, i.e., the leaders may not cross it. We introduce a new drawing style for leaders and with it the problem Triangulated Boundary Labeling. This drawing style guarantees crossing-free drawings even for polygons that have large nonconvexities. We show how to efficiently solve two-sided, three-sided, restricted three-sided, restricted four-sided and four-sided Triangulated Boundary Labeling. Using a fast Matrix Multiplication algorithm and the Min-Plus-Algebra, we can solve Four-Sided Triangulated Boundary Labeling in $\mathcal{O}(n^\omega)$ time, where ω is the *matrix multiplication exponent*. We give some example results of our algorithm and analyse the runtime of an implementation on big instances.

Zusammenfassung

Beschriftungen sind wichtig, um geographische Karten, technische Zeichnungen oder Diagramme zu verstehen. Man kann die Beschriftungen direkt auf dem Bild oder auf einer begrenzenden Linie um das Bild herum plazieren. In diesem Fall werden die wichtigen Elemente der Karte mit sogenannten *leaders* mit der Beschriftung verbunden. Diese können gerade Linien, Kurven oder Polylinien sein. In dieser Arbeit werden die Kanten eines Polygons beschriftet. Das Polygon ist ein *Hindernis*, das heißt, *leaders* dürfen das Polygon nicht durchqueren. Wir führen einen neuen Zeichenstil für *leaders* ein und mit ihm das Problem Triangulated Boundary Labeling. Dieser Zeichenstil garantiert kreuzungsfreie Zeichnungen auch für Polygone, die viele nicht-konvexe Anteile haben. Wir zeigen, wie man effizient zweiseitiges, dreiseitiges, eingeschränktes dreiseitiges, eingeschränktes vierseitiges und vierseitiges Triangulated Boundary Labeling löst. Mittels schneller Matrix-Multiplikation und der Min-Plus-Algebra können wir Four-Sided Triangulated Boundary Labeling in $\mathcal{O}(n^\omega)$ lösen, wobei ω der sogenannte *Matrix-Multiplikations-Exponent* ist. Wir zeigen beispielhafte Ergebnisse unseres Algorithmus' und analysieren die Laufzeit einer Implementierung auf großen Instanzen.

Contents

1 Introduction **4**
1.1 Related Work 7
1.2 Our contribution 8

2 Algorithm for Boundary Labeling using Triangulation **10**
2.1 The Idea 10
2.2 Two-sided labeling 18
2.3 Three-sided labeling with one forbidden outer triangle 20
2.4 Four-sided labeling but leaders may only use outer triangles next to their fan part 21
2.5 Four-sided labeling without restrictions 22

3 Evaluation and Experiments **27**
3.1 Quality 27
3.2 Runtime 30

4 Conclusion and Future Work **35**

Bibliography **36**

1 Introduction

Labels are short descriptions of things that we see on geographical maps or drawings. That can be the name of a street of mountain, some additional information as the address or a rating, or the names of parts in a technical drawing. They are a key part to the understanding geographical maps, pictures or technical drawings. It is important to place them in a way that they are easily readable: The reader should be able to see which part of the map a certain label belongs to. There are some choices that have to be made when labeling a map.

Label position Labels can be placed in different ways. For maps, it is most common to place them directly near the map feature. Figure 1.1a shows a map of the city of Würzburg and the environment. All labels are placed near to or on the corresponding map feature or *site*, for example the names of the city or city parts, street names and the names of mountains or the river. Labels should never overlap.

In Figure 1.1b we see labels that are placed around the figure, in this case a cross-section of the human forearm. The labels are not far away from the features and have about the same shape as the picture. There are lines to connect the labels to the features. These lines are called *leaders*.

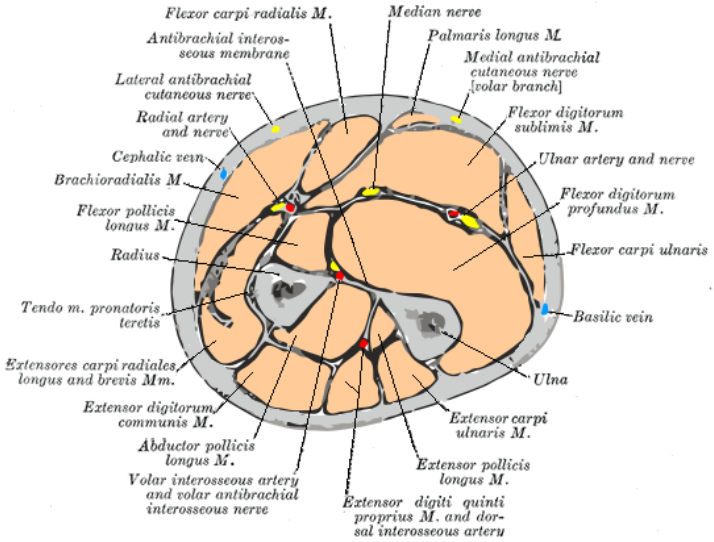
In this thesis, we study *boundary labeling*. Here, the labels are not placed directly on the map but on a bounding box around it. Figures 1.1c and 1.1d give examples for how to use boundary labeling. Figure 1.1c is an example of a building footprint whose outer walls are labeled. This could be used for example to determine how useful every one of the walls is for solar cells, as for certain angles the effectiveness is much higher. Figures 1.1d is an example for a museum that displays different art styles. The labels show which art style is displayed on which wall.

If the labels are only placed on two (three) edges of the bounding box, we call it *two-sided (three-sided) boundary labeling*. Leaders can be placed on adjacent or opposite boundaries.

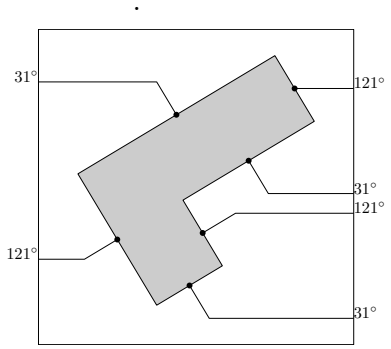
Obstacles Obstacles are part of the drawing area that should not be crossed by leaders. On a geographical map, this could be a important building or some other label that should be well readable and therefore not crossed by leaders or labels. In this thesis we consider the obstacle to be the polygon that we want to label. It must not be crossed by leaders.

Leader style Another choice is the style of the leaders. They can be straight lines (as in Figure 1.1b), curves or polylines. In the case of polylines, it is common constrain

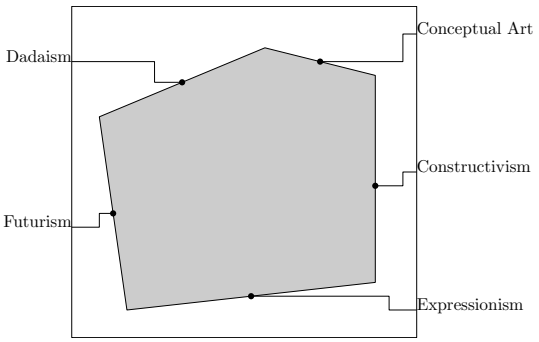
(a) A map of Würzburg and the environment. We can see labels for the whole city, parts of the town, streets as well as the river and some mountains. All labels are directly beside or on the map feature that they describe. ©OpenStreetMap-Contributors, see https://www.openstreetmap.org/copyright



(b) A cross-section through the forearem with labels for the parts [Gra18]



(c) An example of a boundary labeling of a building footprint. The labels give the azimuth of the outer walls. This can be used to evaulate whether solar cells should be placed on this side of the house.



(d) An example museum that exposes different art styles on different walls in the room. The labels indicate which art style is exposed on which wall.

Fig. 1.1: Examples for different labeling styles.

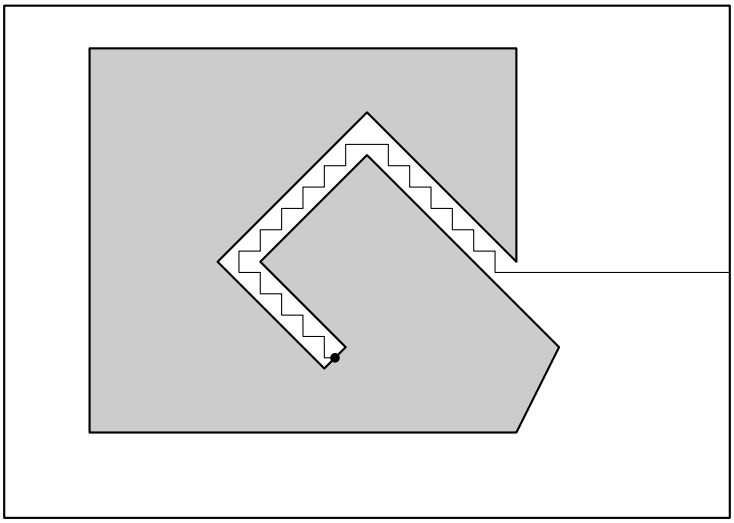


Fig. 1.2: This polygon can not be labeled with type-*opo* leaders withouth them crossing the polygon. Labeling it with leaders that may only be horizontal or vertical makes the leaders have many bends which is neither aesthetically pleasing nor readable.

them to fixed directions. A typical style is the *opo* style. The character *o* stands for *orthogonal* and *p* stands for *parallel*. This means that the first and last parts of the leader is orthogonal to some reference line (typically the side of the image where the labels are placed) whereas the middle part is parallel to the reference. The leaders in Figure 1.1d are drawn in *opo* style. They are orthogonal resp. vertical to the right and left boundaries. In general type-*opo* leaders are very readable and the picture looks good. But we can see in Figure 1.1d that it looks strange when leaders touch the polygon in very acute or very obtuse angles. Also, there are non-convex polygons that can not be labeled by type-*opo* leaders without crossing the boundary of the polygon. An example can be seen in Figure 1.2. It is impossible to label some of the inner sites of the polygon using only type-*opo* leaders, because we would need more than two bends. In this example we can also see that restricting the directions of the leaders (in this case to only horizontal and vertical) can lead to confusing drawings.

In our approach, we will use polylines with arbitrary directions

Fixed or sliding labels In some cases, labels are only placed at fixed points, for example if they should be evenly spreaded around the boundary and on each edge. In this case, the positions of the labels are part of the input. But it can also be part of the problem to find the best position for the labels. *Best* can for example mean that the leaders are as short as possible. It is still important that the labels be readable and do not overlap. In some cases the number of labeled edges of the bounding box is restricted. This means that only one, two or three edges of the bounding box may have labels. Four two-sided labeling, those are typically two opposing edges. If the labels are long, it can be possible to have linebreaks. In our case, the position of the labels is not part of the input. We

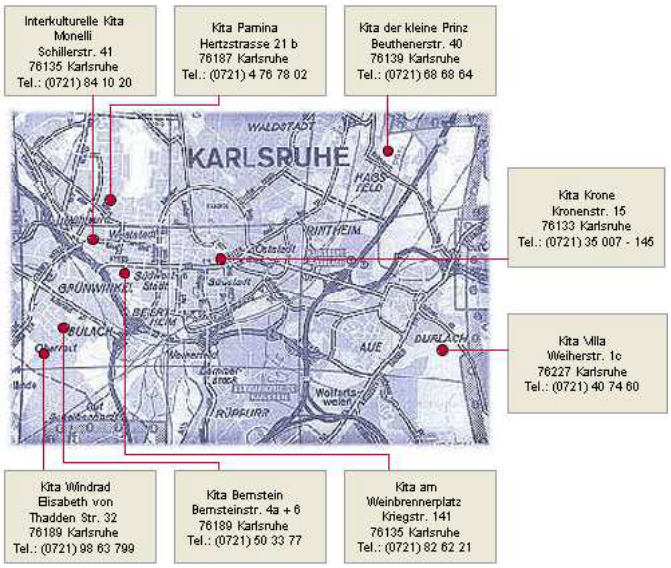


Fig. 1.3: A map of kindergartens in Karlsruhe. The information about them is placed around the maps and there are leaders that connect the information with the kindergartens on the map. This is the first example of boundary labeling [BKSW07].

do not deal with linebreaks, all labels are just written around the bounding box.

1.1 Related Work

An overview about different map labeling problems can be found in the PhD thesis of Claudia C. Iturriaga-Velázquez of 1999 [IV99].

Boundary Labeling was introduced by Bekos et. al. in 2007 [BKSW07]. In their paper, they label a map of kindergartens in the city of Karlsruhe. The information about and adresses of the kindergartens are to be placed around the map and every kindergarten should be connected with its label by a leader. The leaders should not cross, but there are no other obstacles. An example for their output can be found in Figure 1.3. For fixed labels and *opo*-leaders, their algorithm takes $\mathcal{O}(n \log n)$ time to find labeling of minimum total leader length. For straight leaders, they give a time-$\mathcal{O}(n^{2+\varepsilon})$-time algorithm for fixed labels (ε can be arbitrarily small) and a $\mathcal{O}(n^3)$-time algorithm for sliding labels. Bekos et al. also give a recent overview over different labeling techniques [BNN19].

Kindermann et al. show an interesting application for boundary labels [KLW14]: they programmed a LaTeX package that allows to place todo-notes or comments in LaTeX documents. In this case, the labels are the annotations and the sites are the word or paragraphs that should be annotated. They have implemented different leader styles, such as *opo* or curved leaders.

Bose et al. use dynamic programming for boundary labeling with polygonal obstacles [BCK+18]. In their model, the input consists of n points that have to be labeled and polygon obstacles that must not be crossed by the leaders. There are n fixed positions

for the labels on the outer rectangle. They give an algorithm for two-sided labeling with leaders that have at most 1 bend if it exists. The algorithm takes $\mathcal{O}(n^3 \log n)$ time (where n is the number of sites) to find a labeling that minimizes the total leader length if such a labeling exists.

Recently, Bose et al. [BMM21] give algorithms for four three-sided and four-sided labeling with polygonal obstacles. The leaders must have at most one bend. The runtime of these algorithms is $\mathcal{O}(n^3 \log n)$.

The following definition for a constrained Delaunay Triangulation has been taken from a paper by Paul Chew:

Definition 1 (Constrained Delaunay Triangulation [PC89])**.** *Let G be a straight-line planar graph. A triangulation T is a* constrained Delaunay triangulation *(CDT) of G if each edge of G is an edge of T and for each remaining edge e of T there exists a circle c with the following properties:*

1. *The endpoints of edge e are on the boundary of c.*
2. *If any vertex v of G is in the interior of c then it cannot be "seen" from at least one of the endpoints of e (i.e., if you draw the line segments from v to each endpoint of e then at least one of the line segments crosses an edge of G).*

In our case, the planar graph G consists of the vertices and edges of the input polygon and boundary rectangle.

1.2 Our contribution

We define a drawing style that is based on a constrained Delaunay Triangulation of the polygon and bounding box.

Leaders may only have bends on the edges of the triangulation and these bends must be evenly distributed on the edges.

We define the following version of the Map Labeling Problem:

Definition 2 (Triangulated Boundary Labeling)**.**
Input: *A simple polygon $P = [p_1, \dots p_n]$ with $p_i \in \mathbb{R}$ of n vertices and a bounding box $B = [b_1, b_2, b_3, b_4]$ that is a rectangle containing P.*
Output: *A list L of paths that are leaders between the centers of the polygon edges and the edges of B. All corner points of the leaders lie on edges of the constrained Delaunay Triangulation of the input and are evenly distributed on these edges. No leaders cross one another or edges of the polygon. The total length of the leaders is to be minimized.*

In this thesis, we give an algorithm to solve Triangulated Boundary Labeling efficiently. We give several problems that can be solved in $\mathcal{O}(n^2)$ time using our algorithm and show that Four-Sided Triangulated Boundary Labeling can be solved in $\mathcal{O}(n^\omega)$ time, where $\mathcal{O}(n^\omega)$ is the time it takes to multiply two matrices of size $n \times n$ in the min-sum-algebra (*tropical algebra*). As opposed to Bose et al. [BCK+18], this algorithm finds a legal labeling for all instances.

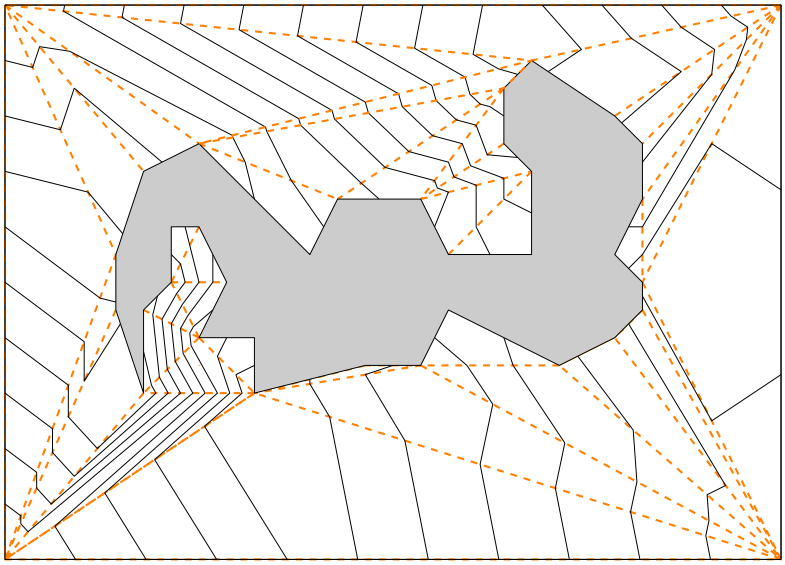


Fig. 1.4: An example for a labeling with our drawing style. The bends of the leaders are only on edges of the triangles. The corner points of the leaders on the triangle edges are evenly distributed.

2 Algorithm for Boundary Labeling using Triangulation

2.1 The Idea

All of the following algorithms use the same dynamic programming idea.

The input (a polygon and a boundary rectangle) is triangulated using a Constrained Delaunay Triangulation. This means that the edges of the polygon and rectangle must be edges in the triangulation. The other edges are elected so that the triangulation is "as much Delauny as possible". A formal definition of the constrained Delaunay Triangualtion can be found in Definition 1. Such a triangulation can be computed in $\mathcal{O}(n \log n)$ time. [PC89].

After having computed the triangulation, we can omit all triangles that lie inside the polygon, as the leaders are only placed outside of the polygon (see Figure 2.1a) and the outer face. Let $\mathcal{T}$ be the triangulation without triangles inside the polygon and the outer face. We will see that the dual graph G_D of $\mathcal{T}$ (the graph where the vertices are the triangles and two vertices are adjacent if the triangles are neighbours) consists of a cycle and some trees (see Figure 2.1b, Theorem 4).

For all other triangles we compute the total leader length of all possibilities how leaders can go through the triangle (see Figure 2.1c for an example of two possibilities in a triangle). Then we combine the triangles to find the shortest total solution using the computations that we already did. See Figure 2.1d for two combined triangles. Here, it is important that we can only add two possibilities that match at the common edge, i.e., if 5 leaders enter one of the triangles, 5 leaders have to leave the other one.

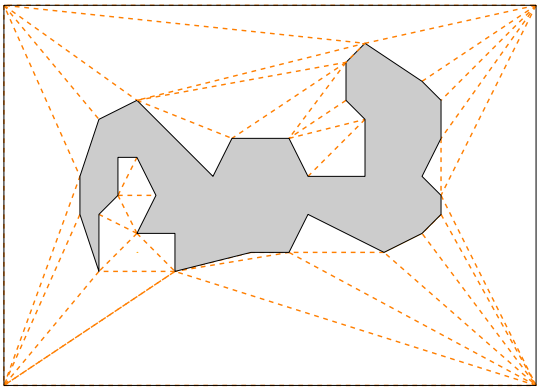
After computing all possibilities for each individual triangle, we select the one with the smallest total length. See Figure 2.1e for a shortest solution.

We begin by showing that the dual of $\mathcal{T}$ has indeed the structure that we use later on.

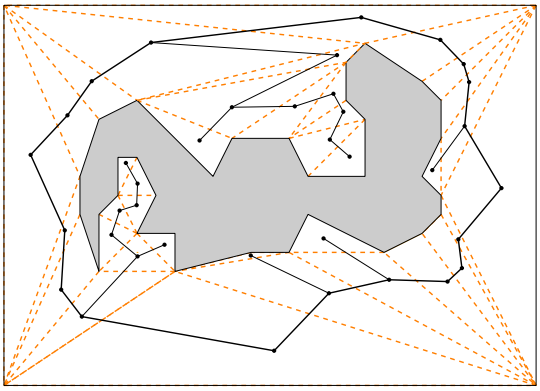
Definition 3 (Pocket)**.** *The area outside of the polygon but inside the convex hull is called a* pocket *or* hole *of the polygon.*

Theorem 4. *The dual graph D_G of $\mathcal{T}$ is a cycle with trees on the inside.*

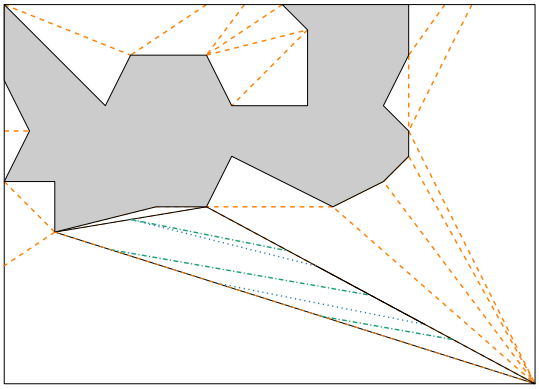
Proof. The bounding box has four corner points. All triangles in $\mathcal{T}$ that share a corner of the bounding box form a path in G_D. The triangles that share an edge with the bounding box are part of two such paths. This means that all the paths are connected and thus form a cycle. If there were two cycles, that would mean that two non-convex parts of the polygon meet and there are thus two polygons. This is a contradiction. □



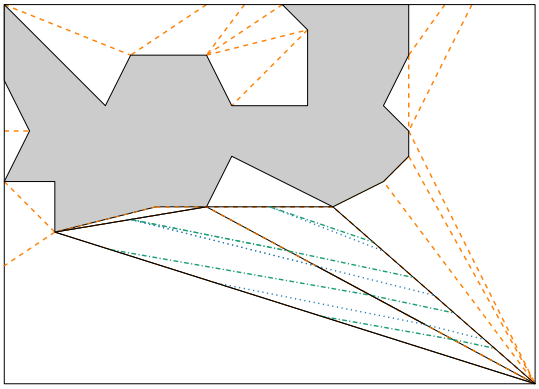
(a) The polygon and its constrained Delaunay triangulation. The edges of the polygon have to be in the triangulation although they may not be part of the real Delaunay triangulation. We consider only the part outside of the polygon.



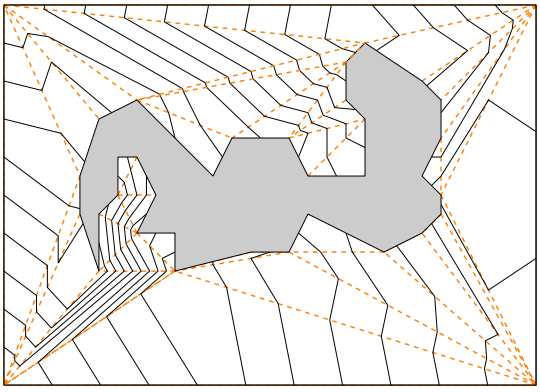
(b) The dual graph G_D of the part of the triangulation that lies outside the polygon. It consists of a cycle and some trees.



(c) Two possibilities for the triangle. In one case, three leaders go through this triangle, in the other case there are only two leaders. We compute the total leader length of all possibilities.

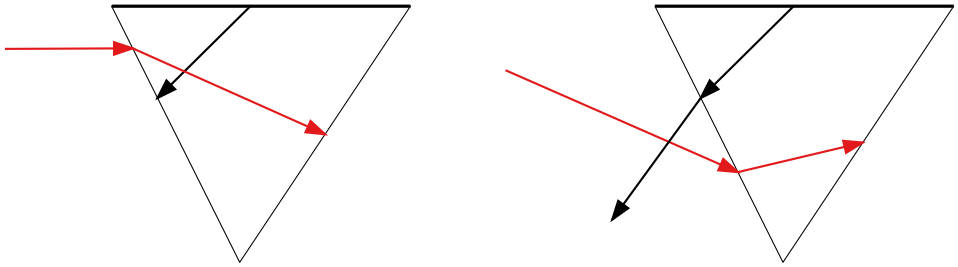


(d) We go through the dual one by one and add more triangles. We have to take care that the number of leaders on common edges is congruent. These are two possibilities for two combined triangles.



(e) The shortest total solution for this polygon and triangulation.

Fig. 2.1: An explanation of our general idea.



(a) The incoming point is closer to the polygon than the outgoing point of the fixed leader.

(b) The incoming point is further away from the polygon than the outgoing point of the fixed leader.

Fig. 2.2: The two cases for a triangle that has incoming and outgoing leaders on the same edge.

Definition 5 (Incoming, outgoing)**.** *We fix the direction of a leader to go from the site to the label and thus have defined the terms* incoming *and* outgoing *leaders on edges of the* $\mathcal{T}$*.*

As already mentioned in Section 1.2, if k leaders enter the triangle in edge A, we divide this edge in $n+1$ parts and distribute the bends of the leaders evenly on the edge. The leaders are not allowed to cross. From this, we get our next lemma.

Lemma 6. *In an optimal solution, every edge of a triangle has either only outgoing or only incoming leaders.*

Proof. All leaders that start at a site or a pocket are clearly incoming, because going into a pocket of the polygon can not result in a shorter leader. Consider the edge where such a leader leaves the triangle. Assume for contradiction there is also an incoming leader at this edge. There are two cases.

Case 1: The incoming leader enters closer to the polygon than the outgoing point of the fixed leader (See Figure 2.2a). This means that the two leader cross inside of the triangle which may not happen.

Case 2: The incoming leader enters further away from the polygon than the outgoing leader (See Figure 2.2b). The incoming leader starts at a site, so following it in the opposite direction leads to the polygon. This means that the outgoing leader that started at a site can never reach the outer rectangle without crossing the incoming leader which leads to a contradiction.

If all leaders leave only at one edge, the other edge can not be an outgoing edge, because there are no more leaders in the triangle that can leave via this edge. □

Definition 7 (Configuration)**.** *Let ABC be a triangle. The (a, b, c) stands fore the number of incoming leaders for edge a resp. b resp. c. It is called a* configuration*. Negative numbers for a, b, c stand for outgoing leaders.*

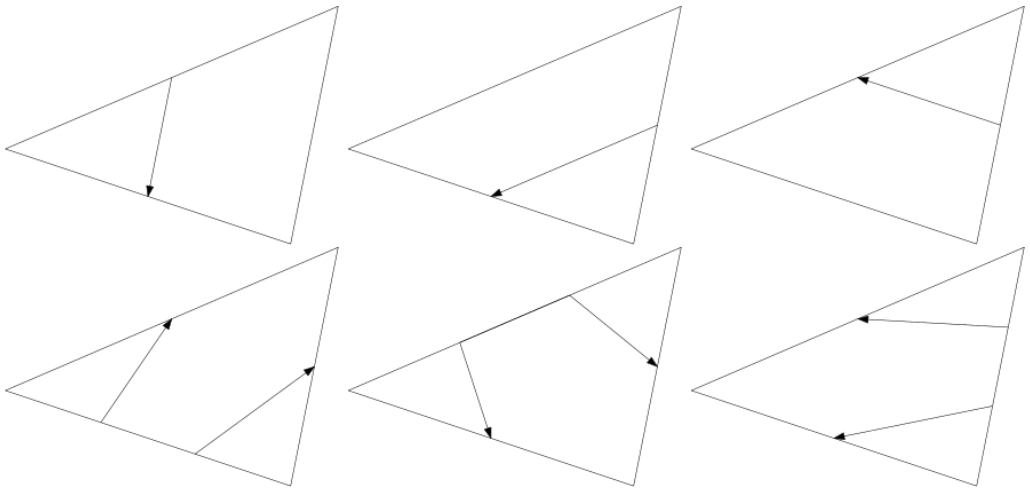
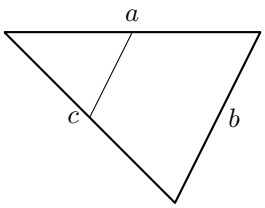
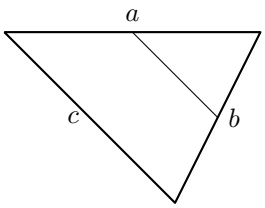


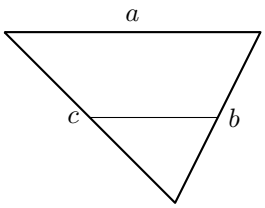
Fig. 2.3: These are the possibilities for leaders in a triangle, where every arrow symbolizes one or more leaders.



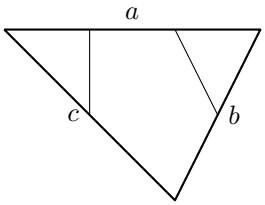
(a) $(1, 0, -1)$ or $(-1, 0, 1)$



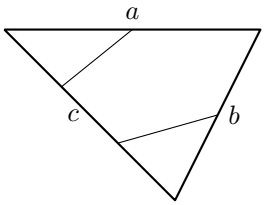
(b) $(1, -1, 0)$ or $(-1, 1, 0)$



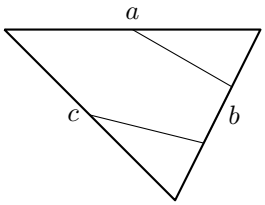
(c) $(0, 1, -1)$ or $(0, -1, 1)$



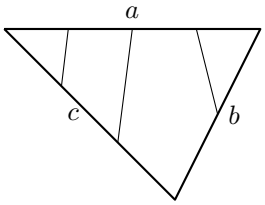
(d) $(2, -1, -1)$ or $(-2, 1, 1)$



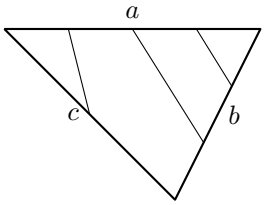
(e) $(1, 1, -2)$ or $(-1, -1, 2)$



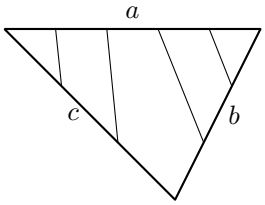
(f) $(1, -2, 1)$ or $(-1, 2, -1)$



(g) $(3, -1, -2)$ or $(-3, 1, 2)$



(h) $(3, -2, -1)$ or $(-3, 2, 1)$



(i) $(4, -2, -2)$ or $(-4, 2, 2)$

Fig. 2.4: Examples how different configurations for one triangle look like.

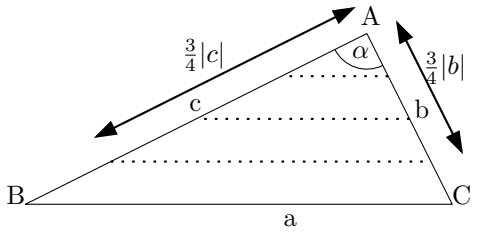


Fig. 2.5: To compute the total length of leaders in the triangle, we compute the length of every single leader via trigonometry and sum them up.

See some examples for configurations in a triangle in Figure 2.4.
It is easy to see that for every configuration (a, b, c)

$$a + b + c = 0$$

holds. This is because we cannot generate or lose leaders.

Cost of a configuration We define the *cost* of a configuration as the total length of leaders in a triangle for the given configuration. We will show that it is possible to compute the cost in constant time for a given configuration.

Lemma 8. *The cost of a configuration for a given triangle can be computed in constant time.*

Proof. We conceptually partition the triangle in t many smaller triangles, where one edge of those smaller triangles is the leader. Its length can now be computed using trigonometry.

For a triangle ABC where α is the angle that lies opposite of edge a, the length of edge a can be computed using the following formula (which is the reformulated cosine formula):

$$|a| = \sqrt{|b|^2 + |c|^2 - 2|b| \cdot |c| \cdot \cos(\alpha)} \tag{2.1}$$

See Figure 2.5 for reference. The total length L of t leaders between the edges b and c of the triangle is

$$L = \sum_{i=0}^{t} \sqrt{\left(\frac{i|b|}{t+1}^2\right) + \left(\frac{i|c|}{t+1}^2\right) - 2\frac{i|b|}{t+1} \cdot \frac{i|c|}{t+1} \cdot \cos\alpha}, \tag{2.2}$$

where $|b|$ resp. $|c|$ denotes the length of edge b resp. c and α is the angle between the edges b and c. We can simplify this formula. First, rearrange some terms.

$$L = \sum_{i=0}^{t} \sqrt{\frac{i^2|b|^2}{(t+1)^2} + \frac{i^2|c|^2}{(t+1)^2} - 2\frac{i^2|b| \cdot |c|}{(t+1)^2} \cdot \cos\alpha} \tag{2.3}$$

Then, we factor out $(t+1)$:

$$L = \sum_{i=0}^{t} \sqrt{\frac{1}{(t+1)^2} \cdot (i^2|b|^2 + i^2|c|^2 - 2i^2|b| \cdot |c| \cdot \cos\alpha)} \tag{2.4}$$

$$L = \sum_{i=0}^{t} \frac{1}{t+1} \sqrt{i^2|b|^2 + i^2|c|^2 - 2i^2|b| \cdot |c| \cdot \cos\alpha} \tag{2.5}$$

As t is not the index of summation, we can take $(t+1)$ out of the sum:

$$L = \frac{1}{t+1} \sum_{i=0}^{t} \sqrt{i^2|b|^2 + i^2|c|^2 - 2i^2|b| \cdot |c| \cdot \cos\alpha} \tag{2.6}$$

We factor out i^2 and take it out of the square root:

$$L = \frac{1}{t+1} \sum_{i=0}^{t} \sqrt{i^2 \left(|b|^2 + |c|^2 - 2bc\cos\alpha\right)} \tag{2.7}$$

$$= \frac{1}{t+1} \sum_{i=0}^{t} i\sqrt{(|b|^2 + |c|^2 - 2|b| \cdot |c|\cos\alpha)} \tag{2.8}$$

As $\sqrt{(|b|^2 + |c|^2 - 2|b| \cdot |c|\cos\alpha)}$ does not contain the index of summation, we can place it in front of the sum and use Gauß's formula for sums:

$$L = \frac{1}{t+1}\sqrt{(|b|^2 + |c|^2 - 2|b| \cdot |c|\cos\alpha)} \sum_{i=0}^{t} i \tag{2.9}$$

$$= \frac{1}{t+1}\sqrt{(|b|^2 + |c|^2 - 2|b| \cdot |c|\cos\alpha)} \frac{t \cdot (t+1)}{2} \tag{2.10}$$

$$= \frac{t}{2} \cdot \sqrt{(|b|^2 + |c|^2 - 2|b| \cdot |c|\cos\alpha)} \tag{2.11}$$

This formula can be computed in constant time, hence configuration of a triangle can be computed in constant time. □

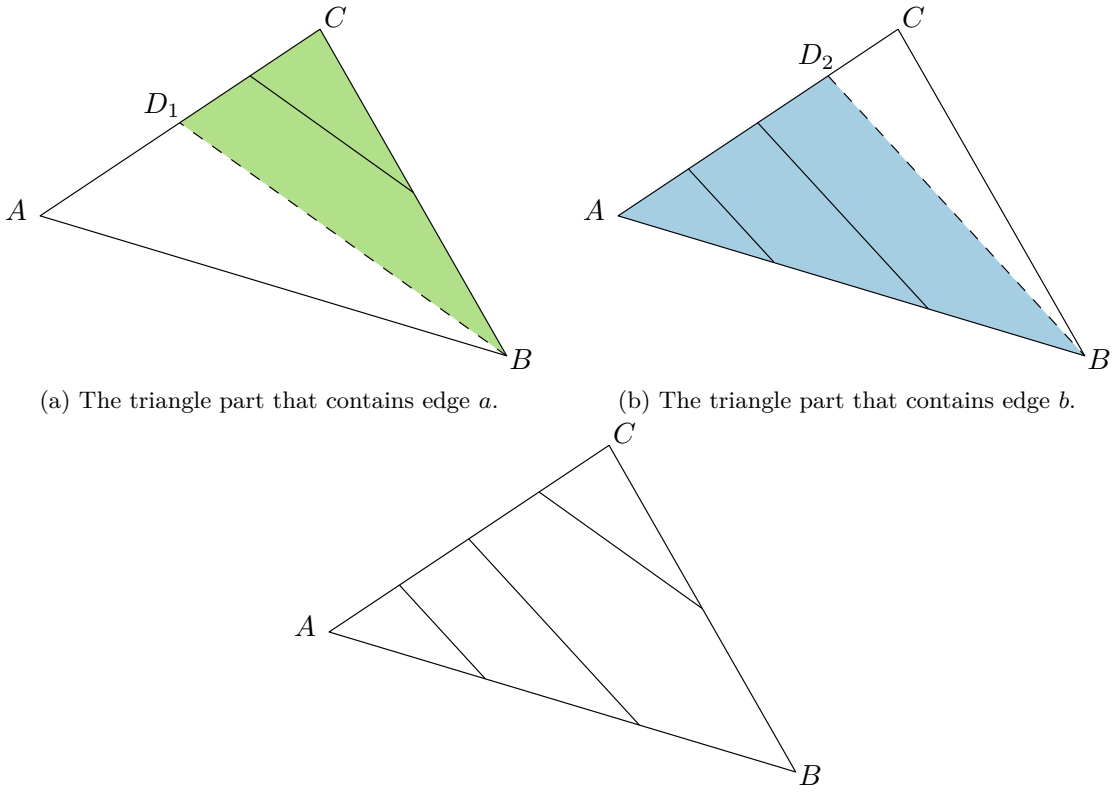
This only considered leaders going straight through the triangle from one edge to another. We can also use this formula for triangles where leaders go through all three edges, see Figure 2.6. For each edge, we split the triangle in two parts. Let ℓ_c, resp. ℓ_a be the number of leader that enter via edge c, resp. edge a. Then, the corner of the new triangle that contains edge c lies on the edge b

$$\frac{\ell_c + 1}{\ell_c + l_a + 1}$$

times the distance $\overline{AC}$ away from point A. The other corner lies

$$\frac{\ell_a + 1}{\ell_c + \ell_a + 1}$$

times the distance $\overline{AC}$ away from point B on edge c.



(a) The triangle part that contains edge a.

(b) The triangle part that contains edge b.

(c) The whole configuration.

Fig. 2.6: To compute the cost of this configuration in triangle a, b, c, split up the triangle in two triangle parts. Now we can compute the length of the leaders in parts a, b, d_2 and b, d_1, c and sum them up.

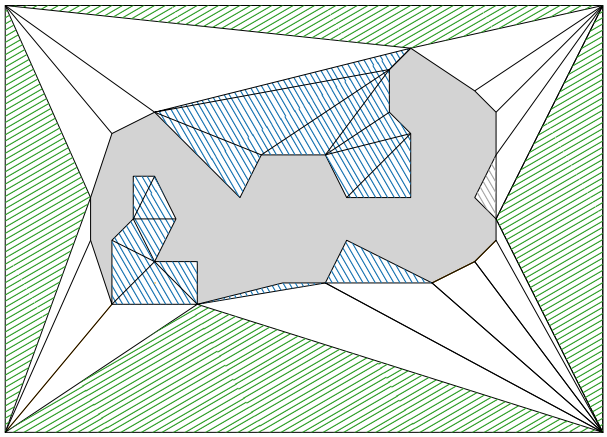


Fig. 2.7: We partition the triangles in three types: The exit triangles (green, rising pattern), the inner triangles (blue, falling pattern) and the fan triangles (white).

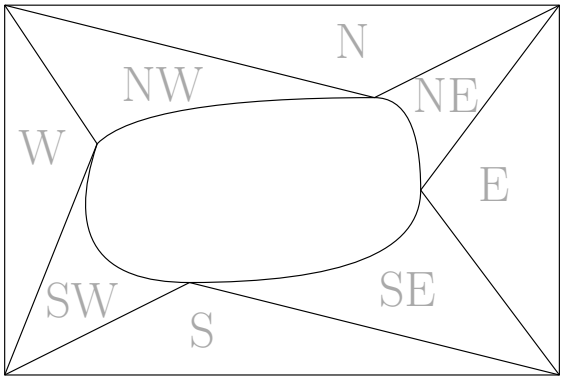


Fig. 2.8: The triangles an corner fan parts are named according to their cardinal direction.

Combining the triangles To get a shortest total solution, we combine all triangles. We distinguish three types of triangles after the triangulation (cf Figure 2.7:

1. triangles in the pockets of the polygon, on the trees of the dual, *inner triangles*
2. triangles that have a common edge with the bounding rectangle, *outer triangles* or *exit triangles*
3. all other triangles, which lie on the cycle of the dual, one endpoint is a corner point of the rectangle, *corner fan triangles*. The set of corner fan triangles that share a particular vertex of the rectangle will be called *corner fan*.

There are four outer triangles in the triangulation (i.e., triangles that have a common edge with the bounding box), which we call N(orth), E(ast), S(outh) and W(est). The corner fans are called NE, SE, SW, NW accordingly (cf. 2.8). The k fan triangles in the NW fan are named $\mathrm{NW}_1, \cdots, \mathrm{NW}_k$, numbered in counterclockwise order.

Lemma 9. *In an optimal solution, there is only one possible configuration for inner triangles. Calculating the total cost of this configurations for all inner triangles takes* $\mathcal{O}(n)$ *time.*

Proof. In an optimal solution, leaders do not enter the pockets, because that can never result in shorter total leaders. So all leaders leave the pockets and since the dual of a pocket is a tree, this is uniquely determined: there is only one way out. This means that there is only one configuration that we need to calculate. We can start from the leaves of the dual graph and one after the other compute the length of the leaders for more triangles. This can be done in time $\mathcal{O}(n)$ for all inner triangles: There are at most n polygon sites that lie inside a pocket. Therefore there are at most $\mathcal{O}(n)$ triangles inside a pocket (because a triangulation of a simple polygon with n vertices has $\mathcal{O}(n)$ triangles). The cost for the configuration of each triangle can be computed in constant time (Lemma 8, so all in all, all costs for the inner triangles can be computed in $\mathcal{O}(n)$ time.

□

All corner fan triangles have on common edge with the polygon or its convex hull. The number of leaders that enter through this edge is therefore determined by the input. We call these *fixed leaders*. For the other edges there are $\mathcal{O}(n)$ possibilities for entering leaders. It is easy to see that two edges determine the number of leaders on the third edge, because as we have seen before the number of incoming leaders minus the number of outgoing leaders is zero. For a corner fan triangle t, by $c_t(i)$, we denote the length of the configuration where i leaders enter the triangle on the edge that comes first in counterclockwise direction.

Lemma 10. *Computing all possible configurations for all corner fans takes $\mathcal{O}(n^2)$ time.*

Proof. Let $t_{c_1,\ldots,c_k}(i)$ denote the cost in all triangles between neighbouring triangles $c_1,\ldots,c_k$ for i leaders that enter c_1 in counterclockwise direction. Let f_i be the number of fixed leaders that directly enter a triangle i from a site or pocket. To sum up a corner fan part c consisting of the triangles $c_1,\ldots,c_k$ we use the following formula:

$$t_c(i) = t_{c_1,\ldots,c_k}(i) = t_{c_1,\ldots,c_k-1}(i) + t_{c_k}(i + (f_1 + \ldots + f_{k-1})) \tag{2.12}$$

$$= t_{c_1}(i) + t_{c_2}(i + f_1) + \ldots + t_{c_k}(i + (f_1 + \ldots + f_{k-1}) \tag{2.13}$$

As we have seen before, the cost of a single configuration can be computed in constant time. We compute the values for $i \in \{-n,\ldots,n\}$. For all four corner fans, there are at most $\mathcal{O}(n)$ summands and we have to compute $\mathcal{O}(n)$ values, which leads to a total time of $\mathcal{O}(n^2)$. □

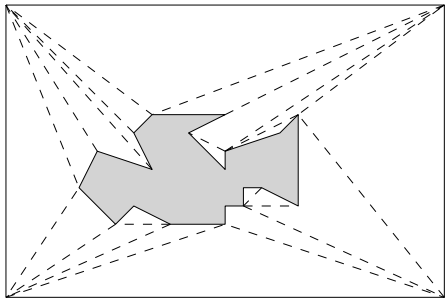
For an outer triangle, there are $\mathcal{O}(n^2)$ possible configurations, because up to n leaders exit the triangle, i.e., their label is at this edge of the rectangle, and n leaders can enter or leave the triangle in one of the other edges of the triangle. So, for an outer triangle e, $c_e(i,j)$ denotes the length of the configuration where i leaders enter the triangle on the edge that comes first in counterclockwise direction and j leaders leave the triangle on the edge that comes last in counterclockwise direction. This results in $i - j$ labels on the outer edge. There are $\mathcal{O}(n^2)$ possible values for each exit triangle. All of them can be computed in $\mathcal{O}(n^2)$ time, because the computation of every configuration takes constant time as seen in Lemma 8.

Undefined values Note that some of the values are undefined. Take for example $c_N(-1,-1)$. There is no possible configuration where leaders only leave an exit triangle. We leave these value as undefined. Whenever we add a number and an undefined value, the result is undefined. The undefined value is greater than any number, so it can not be the minimum of a set.

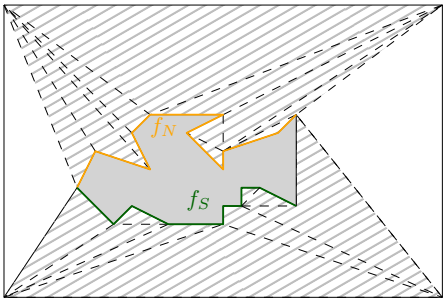
In the following sections we will see how to use this idea for different problems.

2.2 Two-sided labeling

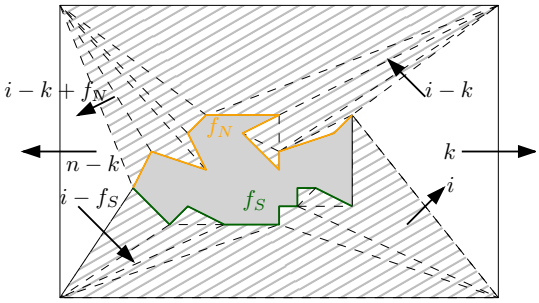
Problem statement For the problem Two-Sided Triangulated Boundary Labeling, only two outer edges of the rectangle may be used to place labels. The edges can be adjacent or opposite.



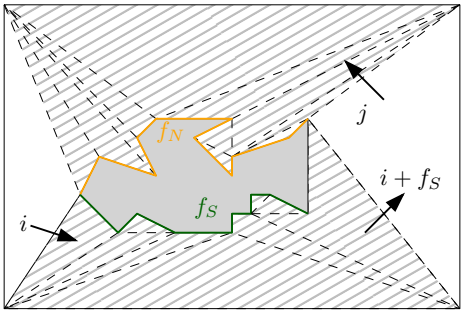
(a) A triangulation of a polygon. We want the labels to be placed on the left and right edge of the rectangle.



(b) First, we combine the fan parts. There are f_1 incoming leaders in the upper part and f_2 leaders in the lower part. We see that $f_1 + f_2 = n$, because the polygon consists of n edges.



(c) To add the last exit triangle, we need to put the two parts together such that the sum of the outgoing leaders sum up to n.



(d) An explanation of Equation 2.15

Fig. 2.9: The steps of the algorithm for two-sided labeling

Theorem 11. Two-Sided Triangulated Boundary Labeling *can be solved in* $\mathcal{O}(n^2)$ *time.*

Proof. See Figure 2.9 for a graphical explanation.

Without loss of generality let the two forbidden triangles be the northern and southern ones. We combine the fan parts as described in Lemma 10 and store the values. In this case, as the northern and southern triangles can't have labels, we can add them as well. Now we have to "fan parts", a southern and a northern one. This means that $c_N(i)$ denotes the total length of the triangles in the northeastern and northwestern fan part (and neighbouring pockets) and the northern exit triangle when i leaders enter the northeastern triangle in counterclockwise direction. By f_N, we denote the number of fixed leaders that enter the northeastern and northwestern corner fan part.

Now, we first add the cost of the eastern and the southern fan part using the following formula.

$$c_{S,E}(i,j) = c_S(i) + c_E(i + f_S, j). \tag{2.14}$$

Then, we add the southern fan part using the following formula.

$$c_{S,E,N}(i,j) = c_{S,E}(i, j - f_N) + c_N(j - f_N) \tag{2.15}$$

Now, $c_{S,E,N}(i,j)$ denotes the cost of the configuration where i leaders enter the northwestern corner fan part in counterclockwise order and j leaders leave the southwestern corner fan part in counterclockwise direction. See Figure 2.9d.

When adding the western exit triangle we compute a vector r for the results. We say that $r(k)$ denotes the length of the smallest configuration where $k \in \{0,\dots,n\}$ leaders leave through the eastern exit triangle and therefore $n-k$ leaders leave through the western exit triangle.

$$r(k) = \min_{i \in \{-n,\dots,n\}} c_{S,E,N}(i - f_S, i - k + f_N) + c_W(i - k + f_N, i - f_S) \tag{2.16}$$

For an explanation of this formula see Figure 2.9c. The computation of this vector takes $\mathcal{O}(n^2)$ time, because we have to sum up $\mathcal{O}(n)$ numbers $\mathcal{O}(n)$ times. The solution is the minimum of the entries in the resulting vector r. □

All of this can be seen in Algorithm 1.

Algorithm 1: Dynamic Program for two-sided labeling

```
1 Compute a Constrained Delaunay Triangulation for the polygon in the rectangle
2 Compute northern and southern fan part
3 Combine eastern and southern costs
4 Combine the result with northern cost
5 Combine with cost of western exit triangle
6 return the smallest entry in the resulting table
```

2.3 Three-sided labeling with one forbidden outer triangle

Problem Statement Restricted Three-Sided Triangulated Boundary Labeling is the special case of three-sided labeling where in one of the outer triangles there must be no leaders. This could be the case, if there is a logo in the lower part of the drawing and there is no space for leaders there.

Theorem 12. Restricted Three-Sided Triangulated Boundary Labeling *can be solved in* $\mathcal{O}(n^2)$ *time.*

Proof. Without loss of generality we assume that the forbidden triangle is the southern one. By $r(j,k)$ we denote the total cost when j labels are placed on the western and k labels are placed on the northern boundary. This means that $n-j-k$ labels must be placed on the eastern boundary, as in total we have to place n labels.

Let $c_{SW,W,NW}(i,j)$ be the cost of the northwestern and southwestern corner fan part and all neighbouring pockets and the western exit triangle when i leaders enter the southwestern fan part and j leaders leave the northwestern fan part in counterclockwise direction. We can precompute and store the values $c_{SW,W,NW}(0,j)$ for all $j \in \{-n,\dots,n\}$

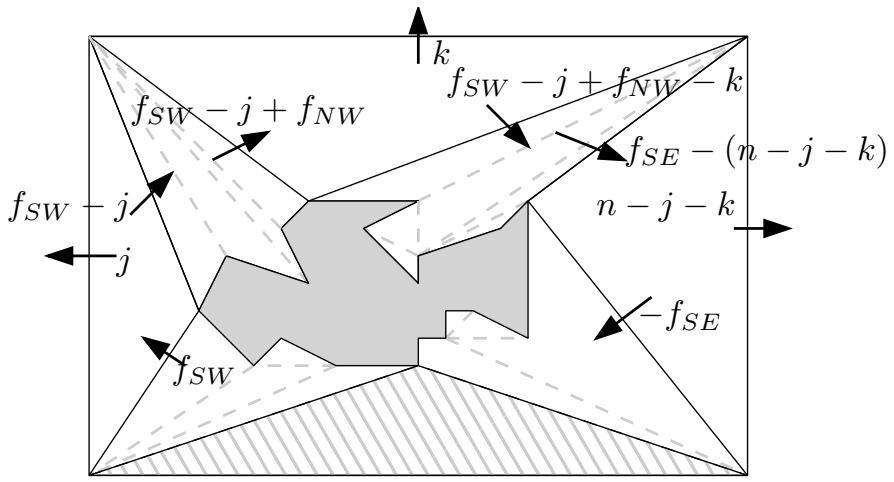


Fig. 2.10: This figure gives an overview over the leaders that enter and leave the various triangles.

in $\mathcal{O}(n^2)$ time. Those are all the values we need for this part, as no leaders may leave the southwestern corner fan part into southern direction.

Let $c_{SE,E,NE}(i,j)$ be defined accordingly for the eastern part. We precompute and store all values $c_{SE,E,NE}(j,0)$ for $j \in \{-n,\ldots,n\}$.

Now,

$$\begin{aligned} r(j,k) = & c_{SW,W,NW}(0, f_{SW} - j + f_{NW} \\ & + c_N(f_{SW} - j + f_{NW}, f_{SW} - j + f_{NW}0 - k) \\ & + c_{NE,E,SW}(f_{SW} - j + f_{NW} - k + f_{NE}, 0) \end{aligned}$$

Confer Figure 2.10 for a graphic explanation of the number of leaders that enter and leave the various parts.

We can compute all values of $r(j,k)$ for $j,k \in \{-n,\ldots,n\}$ in $\mathcal{O}(n^2)$ time. The optimal solution is the smallest entry in $r(j,k)$. □

2.4 Four-sided labeling but leaders may only use outer triangles next to their fan part

Problem Statement For Restricted Four-Sided Triangulated Boundary Labeling, all four edges of the rectangle may be used, but all polygon sites have to be labeled on one of the exit edges that are next to them. We want the smallest solution with smallest total length.

This problem is the most general that we can solve in $\mathcal{O}(n^2)$. A resulting drawing can be more easy to read because the labels are not too far away from the sites they describe.

Theorem 13. Restricted Four-Sided Triangulated Boundary Labeling *can be solved in* $\mathcal{O}(n^2)$*time.*

Proof. For every outer triangle, there are at most $\mathcal{O}(n)$ possibilities as the two neighbouring fans have at most n edges total. We can compute the length of each possibility in $\mathcal{O}(n)$ time because it consists of at most n triangles that need to be combined. Every configuration of an outer triangle can be combined with at most one configuration of a neighbouring outer triangle, so all in all there are only n configurations. We can sum up the total length of every configuration and find the smallest one in $\mathcal{O}(n)$.

All in all this takes $\mathcal{O}(n^2)$ time. □

2.5 Four-sided labeling without restrictions

In this scenario we have four free edges of the rectangle where labels can be placed.

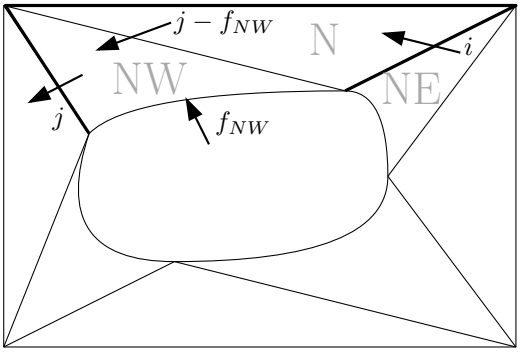
As we have seen before, there are four outer triangles in the triangulation (i.e., triangles that have a common edge with the bounding box), which we call N(orth), E(ast), S(outh) and W(est). The corner fan parts are called NE, SE, SW, NW accordingly (cf. 2.8). The k fan triangles in the NW fan are named $\mathrm{NW}_1, \cdots, \mathrm{NW}_k$, numbered in counterclockwise direction.

We will store the cost of the fan triangle configurations in one-dimensional vectors and the cost of the outer triangle configuration in two-dimensional matrices.

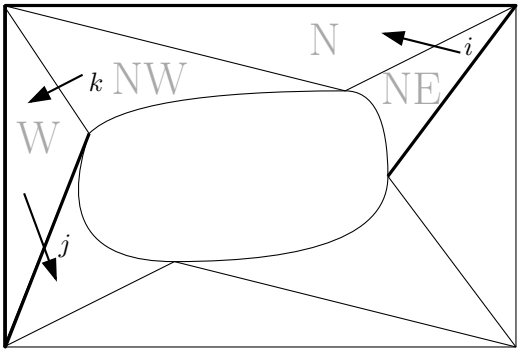
Algorithm 2: Dynamic Program for four-sided labeling

```
1  Compute a Constrained Delaunay Triangulation for the polygon in the rectangle
2  Combine the fan parts for the four exit triangles e                  // O(n^2)
3  do
4  |  for i = −n to n do
5  |  |  for j = −n to n do
6  |  |  |  compute c_e(i, j) and store it
7  Add the northwestern fan part to the northern triangle using the formula in
     Equation 2.17                                                      // O(n^2)
8  Add the western exit triangle using the formula in Equation 2.18    // O(n^ω)
9  Add the south-western fan part
10 Add the southern exit triangle
11 Add south-eastern next fan
12 Add the eastern exit triangle
13 Add the north-eastern fan part
14 return the smallest value in the diagonal of the resulting matrix
```

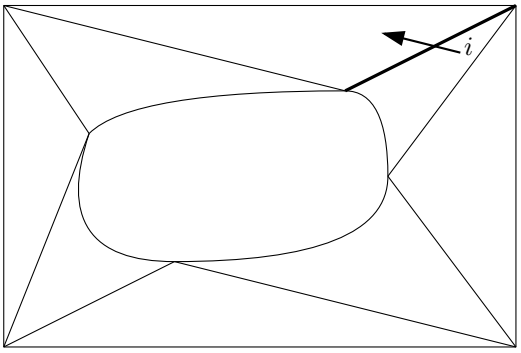
Combining the triangles For an exit triangle e, let $c_e(i, j)$ denote the cost of the configuration where i leaders enter the exit triangle in counterclockwise order and j leaders leave it in counterclockwise direction.



(a) To get the solution for i entering and j leaving leaders, we have to compute $c_N(i, j - f_{NW}) + c_{NW}(j - f_{NW})$.



(b) To get the solution for the additional exit triangle we have to take the minimum over all possible values for k.



(c) When having computed the whole matrix, we can find the solution in the diagonal, because i leaders enter the norther exit triangle and i leaders leave the northeastern fan part.

Fig. 2.11: An explanation of the steps of the algorithm for Four-Sided Triangulated Boundary Labeling.

For a corner fan part t, let $c_t(i)$ denote the cost of the configuration (i.e., the total length of all leaders in the corner fan triangles and the neighbouring pockets) where i leaders enter the corner fan part in counterclockwise direction. Let f_t be the number of fixed leaders that enter the corner fan part from the pockets.

We precompute and store all values $c_t(i)$ for $i \in \{-n, \ldots, n\}$ for the fan triangles.

To get to a solution, we have to combine the exit triangles with the corner fan parts. We start with the northern exit triangle and combine it with the northwestern fan part. Let $c_{N,NW}(i,j)$ be the cost of the configurations in the northern exit triangle and the northwestern fan part combined when i leaders enter the northwester corner fan part and j leader leave the northern exit triangle in counterclockwise direction.

To compute $c_{N,NW}(i,j)$, we use the following formula.

$$c_{N,NW}(i,j) = c_N(i, j - f_{NW}) + c_{NW}(j - f_{NW}) \tag{2.17}$$

We have to add the value $c_{NW}(j - f_{NW})$ because f_{NW} additional leaders start in the fan part which means that $j - f_{NW} + f_{NW} = j$ leaders leave it. For a graphical explanation, confer Figure 2.11a.

As a next step, we add the next exit triangle to get the resulting matrix $c_{N,NW,W}$. Note that $c_{N,NW,W}(i,j)$ denotes the total length of the leaders in the northern and western extit triangle, the northwestern corner fan part and all its neighbouring pockets when i leaders enter the western exit triangle and j leaders leave the northern exit triangle in counterclockwise direction. We use the following formula

$$c_{N,W}(i,j) = \min_{-n<k<n} c_{N,NW}(i,k) + c_W(k,j) \tag{2.18}$$

This formula is explained in Figure 2.11b. We compute all legal solutions, i.e. solutions where the number of leaving leaders for the first part equals the number of incoming leaders for the second part. Then we take the minimum.

We add all other parts in the same way, using formulas according to those in Equations 2.17 and 2.18. The result is a matrix C where $C(i,j)$ denotes the total length of all leaders when i leaders enter the northern exit triangle from the north-eastern fan part and j leaders leave the north-eastern fan part into the northern exit triangle (cf. Figure 2.11c). So it is evident that actual solutions are only found in the diagonal of the matrix. Therefore we have to search the diagonal for the smallest solution. This gives us the number of leaders that have to cross between the north-eastern fan part and the northern exit triangle to get the smallest possible leader length.

Finding the actual drawing So far, we only computed the minimum sum of the leaders. But to draw the labeling we need the exact number of leaders that go through every edge. We can find it by going backwards through our computation and see which numbers realize the minimum. This can be done in $\mathcal{O}(n^2)$ time.

Runtime (simplified analysis)

Theorem 14. *Four-sided labeling without restrictions can be solved in $\mathcal{O}(n^3)$ time.*

Proof. As explained in Section 2.1 the computation for the number and length of all leaders in the inner triangles can be done in $\mathcal{O}(n)$ time, computing all vectors for the fan triangles and combining them to fans takes $\mathcal{O}(n^2)$ time and computing the matrices for the exit triangles takes $\mathcal{O}(n^2)$ time.

Adding a fan triangle as it is shown in Equation 2.17 can be done in $\mathcal{O}(n^2)$ time, because we have to do a simple addition for every of the resulting $\mathcal{O}(n^2)$ matrix cells.

Adding an exit triangle takes more time. For every resulting matrix cell, we have to find the minimum over $\mathcal{O}(n)$ sums. This can be done in $\mathcal{O}(n^3)$ time, because we compare $\mathcal{O}(n)$ number for each of the $\mathcal{O}(n^2)$ resulting cells. Finding the value in the last matrix can be done in $\mathcal{O}(n)$ time, because we only have to look at the values on the diagonal.

As shown in the preceeding paragraph, the actual values for all leaders can be found in $\mathcal{O}(n^2)$ time.

□

Tropical algebra In order to improve the runtime, in particular the combination of the exit triangles, we consider the so-called *tropical algebra*. The following definitions are taken from [Lit13].

Definition 15 (Semiring)**.** *Let $(S, \oplus, \otimes)$ be a tuple of a set and two operations. The tuple $(S, \oplus, \otimes)$ is a* semiring *if*

- *$\oplus$ and $\otimes$ are associative,*
- *$\oplus$ is commutative,*
- *$\otimes$ is distributive with respect to $\oplus$.*

An example for a semiring is $(\mathbb{R}, +, \cdot)$. From this example, we can see why the symbols that are normally used to define a semi-ring are close to $\cdot$ and $+$. But we can also define rings with other operations.

Definition 16 (Tropical Algebra)**.** *The* tropical algebra *(or* min-plus algebra*) is the semiring $(\mathbb{R} \cup \{\infty\}, \oplus, \otimes)$, where*

$$x \oplus y = \min(x, y) \text{ for } x, y \in \mathbb{R} \cup \{\infty\}$$

$$x \otimes y = x + y \text{ for } x, y \in \mathbb{R} \cup \{\infty\}$$

The tropical algebra got is name in honour of the Brazilian mathematician Imre Simon who was one of the pioneers in this field [Pin98].

Multiplying matrices The time needed to multiply two $n \times n$-matrices is typically denoted by $\mathcal{O}(n^\omega)$. We call ω the *matrix multiplication exponent*. Clearly, $2 \leq \omega \leq 3$, because the naive way of computing every entry of the resulting matrix by mutliplying and adding the entries of the matrices takes $\mathcal{O}(n^3)$ many steps and we need to at least write the $\mathcal{O}(n^2)$ entries into the resulting matrix.

Only in 1969, Volker Strassen published an algorithm showing $\omega < 2.8$ [Str69]. This algorithm uses a divide-and-conquer approach to reach this low runtime. Since then, the bound has been lowered several times. In 1991, Coppersmith and Winograd lowered to bound to $\omega < 2.376$ [CW90]. Their algorithm also partitions the matrix in several smaller matrices and uses recursion. Since then, this algorithm as well as the runtime analysis has been improved.

As of 2020, the matrix multiplication exponent is proven to be $\omega < 2.373$ [Wil12].

All known algorithms for matrix multiplication do not specify which algebra has to be used. So we can use the same algorithm for the tropical algebra. Let A and B be two $n \times n$-matrices.

Recall the formula for matrix multiplication. Normally, it is

$$c_{i,j} = \sum_{s=1}^{n} a_{i,s} \cdot b_{s,j}$$

But multiplying those matrices in the tropical algebra results in a $n \times n$-matrix C where

$$c_{i,j} = \min_{1 \leq s \leq n} a_{i,s} + b_{s,j}$$

This is the operation that we need to compute Equation 2.18.

Runtime (improved analysis)

Theorem 17. Four-Sided Triangulated Boundary Labeling *can be solved in* $\mathcal{O}(n^\omega)$ *time.*

Proof. The inalysis in Theorem 14 showed that all operations but the combination of another exit triangle can be done in $\mathcal{O}(n^2)$ time. As we have just seen, this combination can be done in $\mathcal{O}(n^\omega)$ time.

We know that $\mathcal{O}(n^2) \subseteq \mathcal{O}(n^\omega)$, so the whole runtime of the algorithm is $\mathcal{O}(n^\omega)$. □

3 Evaluation and Experiments

In the following, we will qualitatively evaluate the drawings by our algorithm using criteria of Bekos et al. [BNN19]. Afterwards, we evaluate the runtime of an implementation on big instances.

3.1 Quality

Bekos et al. [BNN19] give an overview over different labeling techniques and algorithms. To evaluate them, they define the following criteria for a good labeling:

C1 The leaders have small length.

C2 The number of leader crossings and label overlaps is small.

C3 The labels mimic the shape of the image.

C4 The labels are distributed evenly.

C5 The directions of the leader segments comply with a set of preferred directions.

C6 The leaders have a small number of bends.

C7 There is sufficient space between leaders.

C8 Labels consist of single text lines if possible.

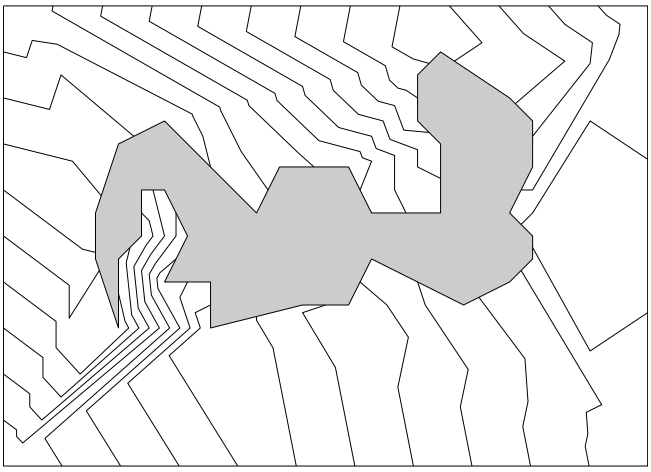
C9 Labels that are semantically related are grouped.

Some of the criteria like C7 are subjective, others, such as C4, can be measured. Criteria like C1 and C5 or C1 and C3 can contradict each other. Some others, like C1 and C6, may influence each other. So in some cases, trade offs are needed.

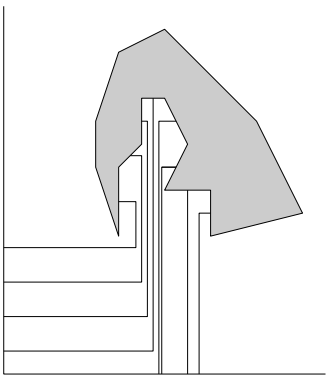
In our context, C3, C8, and C9 are not relevant. The labels can not mimic the shape of the image as the bounding box is part of the input. We also do not arrange the labels nor care about their length or content.

We will have a look at the other criteria one by one to see to which extent the outputs of our algorithm fulfil them. An example output can be found in Figure 3.1a.

C1 The solution has the smallest length of all possible solutions for Triangulated Map Labeling, but we can see that it would be easy to shorten most of the leaders by omitting superfluous bends and thus introducing shortcuts. To do this, we have to drop our drawing style.

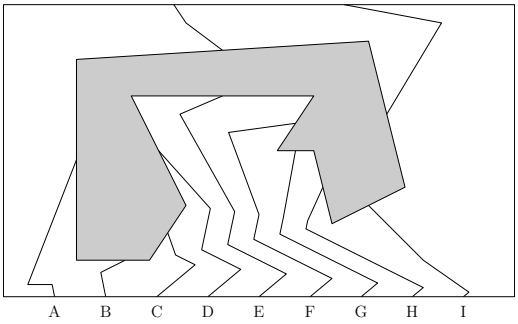


(a) An example of a four-sided labeling of a polygon. We can see that the leaders are evenly placed. There are some unneccesary bends. The drawing area is well used.

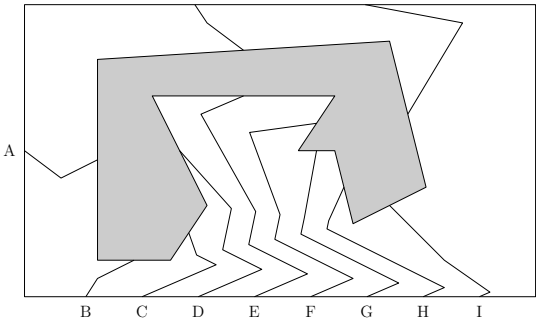


(b) A hand drawn labeling of the lower left part of the same polygon and same bounding box, labeled with *opo*-leaders. Because there is less space between the leaders and they are parallel, the leaders are more difficult to distinguish.

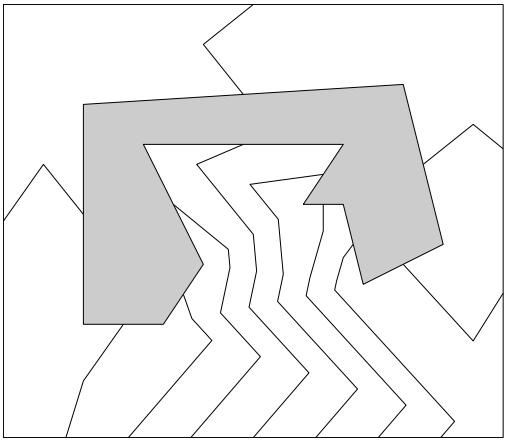
Fig. 3.1: An example of a polygon labeled with our algorithm and a hand-drawn alternative.



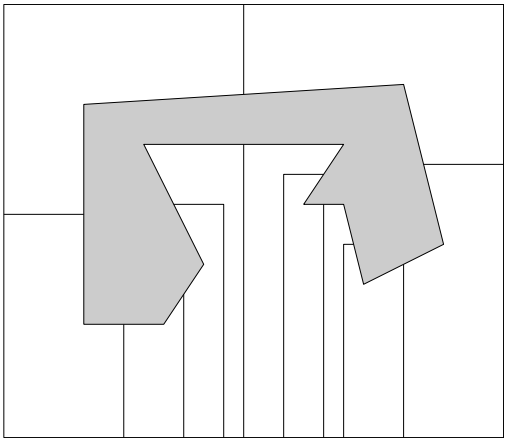
(a) This is the optimal solution for the input. There are no labels at all on the left and right boundary. This looks unbalanced and it may not be intuitive that the leader on the left site of the polygon does not simply go straight to the left.



(b) In this case the leader on the left site of the polygon goes directly to the left boundary. This leads to longer leader that go downwards, because there endpoints on the lower edge more to the left. This version has fewer bends.



(c) This is the same polygon with a slightly bigger bounding box. We can see that this changes the output drastically.



(d) A hand-designed axis-parallel version with as few bends as possible. The leaders are not evenly positioned over the rectangle edges and the parallel leaders can not be distinguished easily.

Fig. 3.2: The same polygon, labeled in four different versions. We can see how the bounding box influences the result.

C2 This criterion is fulfilled as we do not have any crossings.

C4 The labels are distributed evenly on their edges. However, it can be that some edges do not have any labels at all (for an example, see Figure 3.4c). The algorithm could be changed such that the maximum or minimum number of leaders on each boundary edge is constrained. This can be done by returning *undefined* for exit triangle configurations that result in too many or too few labels. This can be used to balance the number of labels on the boundary rectangle.

C5 We do not restrict the directions of the leaders, so they can take all possible directions. While this contradicts the criterion, it makes the image far more readable, especially in the presence of nonconvexities in the polygon. We can see this in Figure 3.1. Figure 3.1a has non-parallel leaders inside the pockets. The drawing space inside the pockets is well used, the leaders are distributed evenly. Outside of the pockets, were the leaders are more or less parallel, they are more difficult to distinguish. In Figure 3.1b, we use *opo*-leaders to label the same part of the polygon by hand. Those leaders cannot be evenly distributed, because they must not have more than two bends. The leaders are parallel and therefore not that well distinguishable. This version is worse to read, although its leaders have less different directions.

C6 The number of bends for a leader equals the number of triangles that it has to pass before it reaches the boundary box. It is not restricted and we can see that there are many "unnecessary" bounds. This is a downside of the problem statement. We could do postprocessing to omit some of the bends.

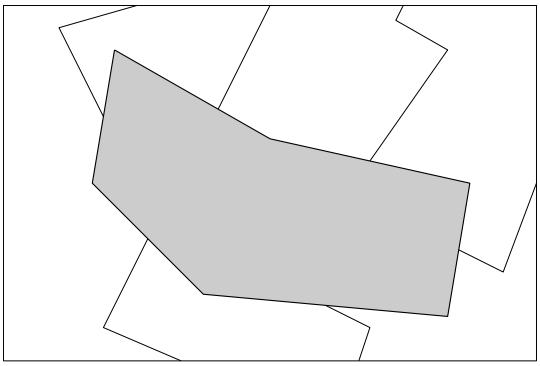
C7 Almost all space of the image is used and the leaders are distributed rather evenly in the whole image. Therefore the leaders have enough space between them wherever possible.

The output for small, convex inputs or inputs where most of the vertices lie on the convex hull tends to have an excessive number of bends. Those instances should be labeled using *opo*-leaders or straight lines. We can see this in Figure 3.3. Our algorithm gives a labeling that has more bends than needed. We can just label it using straight lines. The *opo*-style labeling still has unnecessary bends, but looks nicer than our result. The version with evenly distributed labels and straight lines looks worst.

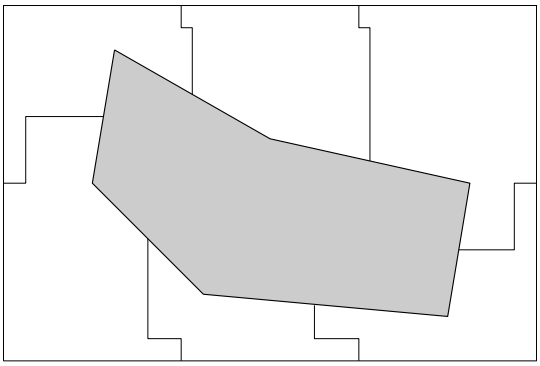
But for images with big pockets like Figure 3.1a, where *opo*-leaders do not work, our algorithm gives good results. It could be beneficial to combine both approaches – using our triangulated algorithm for the pockets and *opo*-stye leaders (or even straight-line leaders) for the part outside of the convex hull.

3.2 Runtime

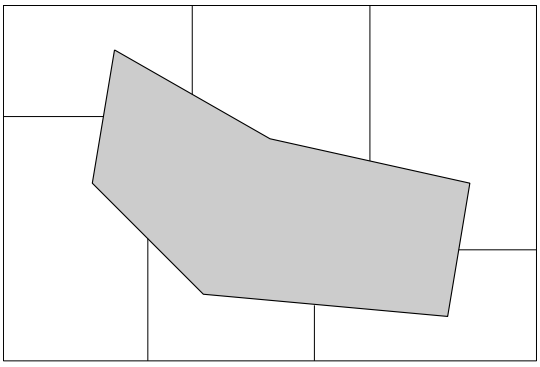
The algorithm for four-sided labeling was implemented in Java. As input, it takes a svg file containing two paths: a rectangle (the bounding box) and a polygon that lies inside



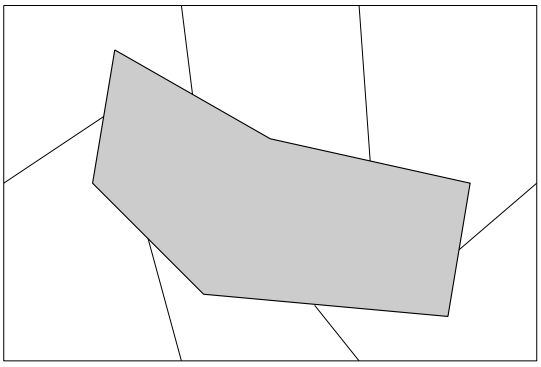
(a) This polygon is labeled using our algorithm. There are unneeded bends.



(b) This polygon is labeled using *opo*-type leaders. There are more bends than in the first version.



(c) In this version, we use only horizontal or vertical leaders. They go directly from the polygon site to the bounding edge.



(d) This version uses straight lines. The labels are evenly distributed on the bounding edge and the leaders are just straight lines from the polygon sites to the labels.

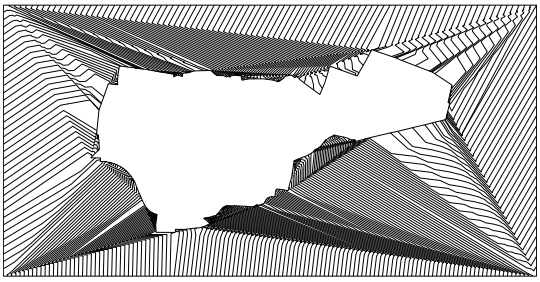
Fig. 3.3: Four different version of a four-sided labeling for this small polygon.

the bounding box. It outputs an svg or ipe file that contains one path for the bounding box, one for the polygon and one per leader.

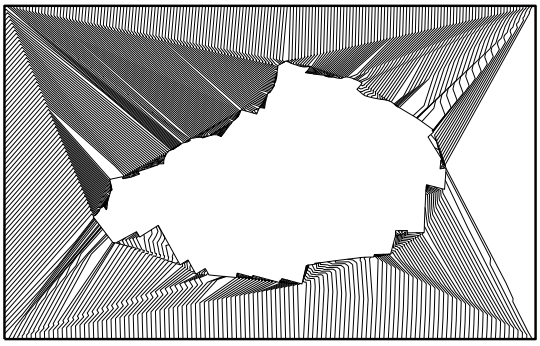
We tested the runtime on five polygons that were obtained from Open Street Maps. [1] We have the municipal boundaries of Gerbrunn (279 vertices), the island Amrum (301 vertices), the city of Veitshöchheim (392 vertices), the lake Brombachsee (482 vertices) and the city of Kürnach (350 vertices). We ran the algorithm five times and took the average runtime. The results can be seen in Figure 3.5. Most of the time is used to combine the matrices for the exit triangles and the fans. This is not surprising, as we did not implement fast matrix multiplication, but used a self-implemented brute-force method which takes $\mathcal{O}(n^3)$ time. Even by using Matrix libraries we could most probably improve the runtime. Also, Huang et al. [HSHvdG16] showed that the Strassen Algorithm is efficient in practice even on small matrices. Since it is the bottleneck, using a faster way to compute Matrix Multiplication on the min-sum-algebra (tropical algebra) could greatly reduce the runtime.

Looking at the pictures of the polygons with leaders we can see that a picture with this many leaders is actually no longer readable. This means that such big instances are not a realistic use case and therefore the long runtime is not a problem. For instances of realistic size, our algorithm takes less then 10 seconds.

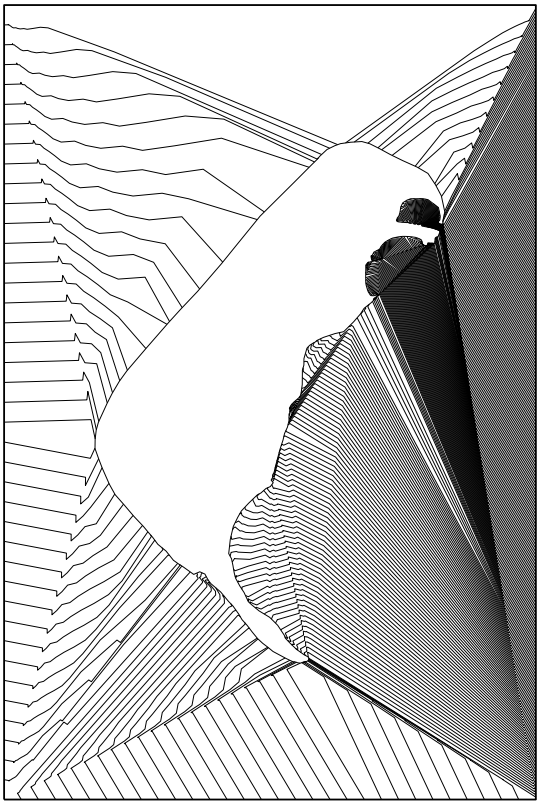
[1] www.openstreetmaps.org



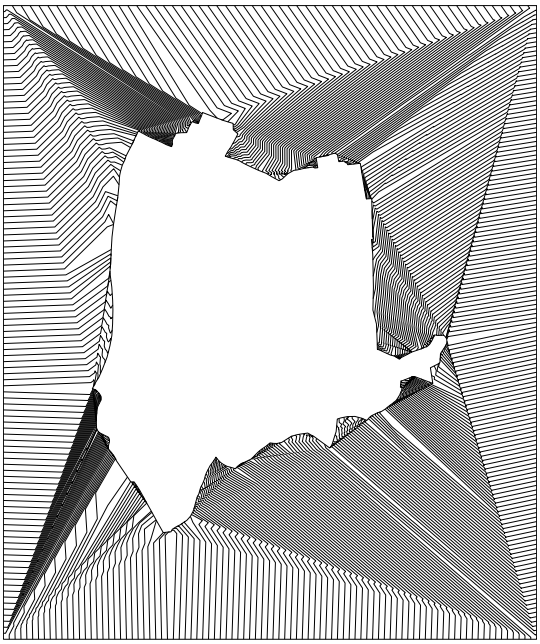
(a) Four-sided labeling of Gerbrunn



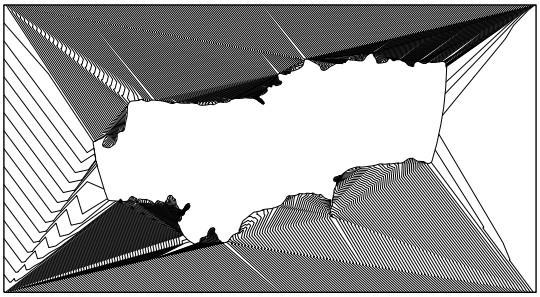
(b) Four-sided labeling of Kürnach



(c) Four-sided labeling of the island of Amrum



(d) Four-sided labeling of Veitshöchheim



(e) Four-sided labeling of the lake Brombachsee

Fig. 3.4: Five big instances labeled by our algorithm.

name	number of vertices	total time
Gerbrunn	279	14.86s
Amrum	301	25.61s
Kürnach	350	36.39s
Veitshöchheim	392	53.93s
Brombachsee	482	103.99s

Fig. 3.5: The runtime for instances of different size.

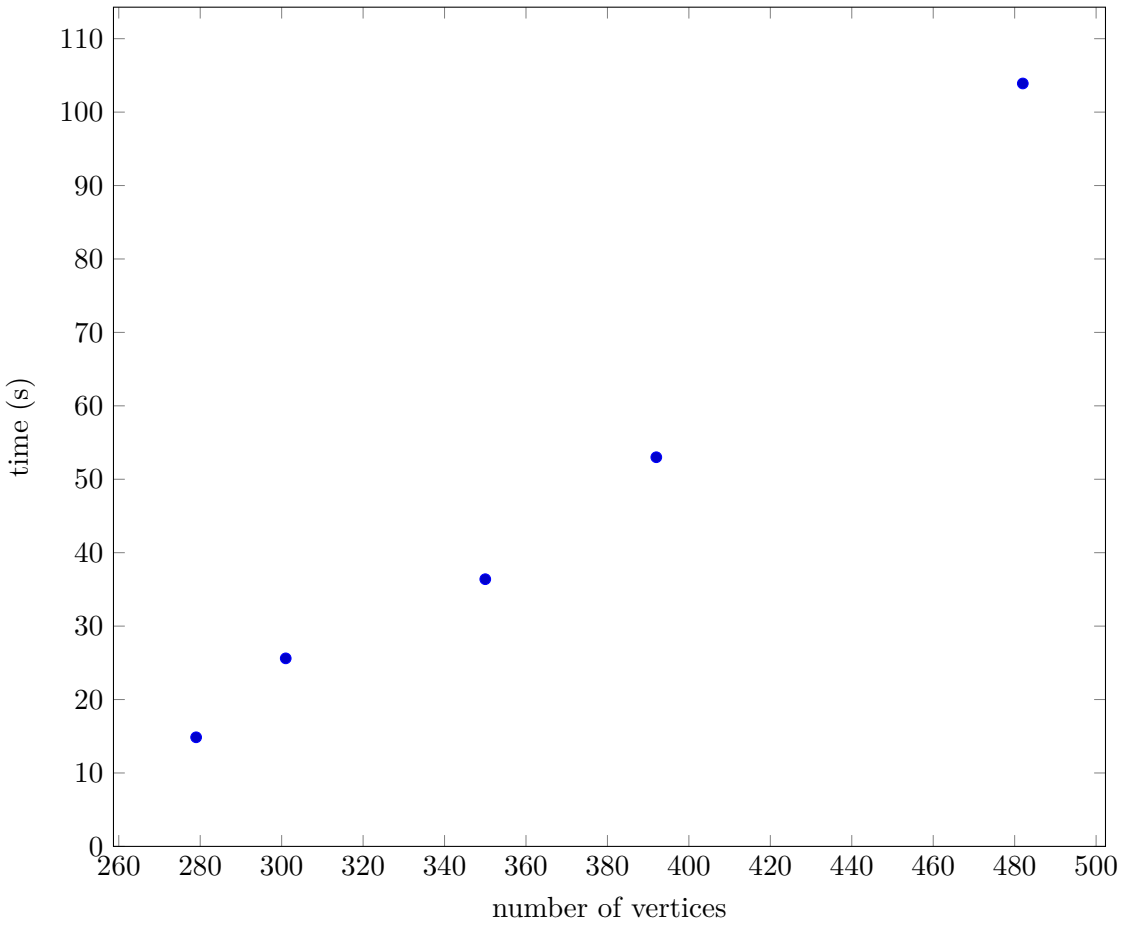


Fig. 3.6: The runtime of our algorithm on instances with many vertices.

4 Conclusion and Future Work

There are many approaches to labeling maps. In this thesis, we introduce Triangulated Boundary Labeling. In Chapter 2.1, we explained the idea of a dynamic program to find a length-minimal solution. Then we gave different problem statements and showed how to solve them using our approach. We have shown that Four-Sided Triangulated Boundary Labeling can be solved in $\mathcal{O}(n^\omega)$ time, i.e., the time for matrix multiplication on the min-sum-algebra (tropical algebra). We showed that 2-sided Triangulated Boundary Labeling, Restricted 3-sided Triangulated Boundary Labeling and Restricted 4-sided Triangulated Boundary Labeling can be solved in $\mathcal{O}(n^2)$ time.

In Chapter 3, we gave some example results for polygons of different sizes and shapes and evaluated the actual runtime for bigger inputs. We have seen that our algorithm has some advantages, especially for labeling non-convex polygons with large pockets.

Of course, our idea and algorithm can also be used for maps that are not bounded by a rectangle, but by other shapes. As we need the dual graph of the triangulation to be a cycle with trees, it only works for shapes that guarantee this structure. As long as the bounding polygon has constant size, this does not change the runtime.

In some cases, not all of the polygon sites should get labels. If only k polygon sites are labeled, we get the following runtimes with our approach: $\mathcal{O}(n \log n)$ for the constrained Delaunay triangulation, $\mathcal{O}(n)$ for the pockets, $\mathcal{O}(k)$ for the computing of all configurations of the corner fan triangles, $\mathcal{O}(n \cdot k)$ for combining all corner fan parts. $\mathcal{O}(k^2)$ for computing the configurations of the exit triangles, $\mathcal{O}(k^2)$ for adding the corner fans and $\mathcal{O}(k^\omega)$ for combining the exit triangles. This means that if the triangulation is already part of the input, we get a total runtime of $\mathcal{O}(n + k^\omega)$, for a constant k the algorithm has linear runtime in n. If the triangulation is not part of the input, the total runtime is $\mathcal{O}(n \log n + k^\omega)$, for a constant k it is $\mathcal{O}(n \log n)$.

As we have seen in Section 3, there are many bends in the leaders. Some of them are not needed to prevent crossings with other leaders or the polygon but arise only from our drawing style. One could get rid of superfluous bends, for example using curve simplification by Dyken et al. [DDS09]. They give an algorithm that simplifies curves while preserving the topology, so without introducing new crossings.

Another idea to minimize the bends is to use the triangulated approach only inside the polygon pockets, where we can not garantuee to find a solution for *opo*-leaders. The convex hull can then be labeled using *opo*-leaders. In total this results in fewer bends than our algorithm and in legal labelings for all instances.

As described earlier, we could easily change the algorithm such that we limit the number of labels that can be placed on each edge of the rectangle respectively. This leads to a more balanced placement of the labels.

Bibliography

[BCK+18] Prosenjit Bose, Paz Carmi, J. Mark Keil, Saeed Mehrabi, and Debajyoti Mondal: Boundary Labeling for Rectangular Diagrams. In David Eppstein (editor): *16th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2018)*, volume 101 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[BDLN02] Carla Binucci, Walter Didimo, Giuseppe Liotta, and Maddalena Nonato: Labeling Heuristics for Orthogonal Drawings. In Petra Mutzel, Michael Jünger, and Sebastian Leipert (editors): *Graph Drawing 2001 (GD'01)*, volume 2265 of *Lect. Notes Comput. Sci.*, pages 139–153. Springer-Verlag, 2002.

[BDLN05] Carla Binucci, Walter Didimo, Giuseppe Liotta, and Maddalena Nonato: Orthogonal Drawings of Graphs with Vertex and Edge Labels. *Computational Geometry: Theory & Applications*, 32(2):71–114, 2005.

[BKNS10] Michael A. Bekos, Michael Kaufmann, Martin Nöllenburg, and Antonios Symvonis: Boundary Labeling with Octilinear Leaders. *Algorithmica*, 57(3):436–461, 2010.

[BKSW07] Michael A. Bekos, Michael Kaufmann, Antonios Symvonis, and Alexander Wolff: Boundary labeling: Models and efficient algorithms for rectangular maps. *Computational Geometry*, 36(3):215–236, 2007.

[BMM21] Prosenjit Bose, Saeed Mehrabi, and Debajyoti Mondal: Faster Multi-sided One-Bend Boundary Labelling. In Ryuhei Uehara, Seok-Hee Hong, and Subhas C. Nandy (editors): *WALCOM: Algorithms and Computation - 15th International Conference and Workshops, WALCOM 2021, Yangon, Myanmar, February 28 - March 2, 2021, Proceedings*, volume 12635 of *Lecture Notes in Computer Science*, pages 116–128. Springer, 2021.

[BNN19] Michael A. Bekos, Benjamin Niedermann, and Martin Nöllenburg: External Labeling Techniques: A Taxonomy and Survey. *Computer Graphics Forum*, 38(3):833–860, 2019.

[CW90] Don Coppersmith and Shmuel Winograd: Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990. Computational algebraic complexity editorial.

[DDS09] Christopher Dyken, Morten Dæhlen, and Thomas Sevaldrud: Simultaneous curve simplification. *Journal of Geographical Systems volume*, 11:273–289, 2009.

[Gra18] Henry Gray: *Anatomy of the Human Body*. Lea and Febiger, 1918.

[HSHvdG16] Jianyu Huang, Tyler M. Smith, Greg M. Henry, and Robert A. van de Geijn: Strassen's Algorithm Reloaded. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '16. IEEE Press, 2016.

[IV99] Claudia C. Iturriaga-Velazquez: *Map labeling problems*. PhD thesis, University of Waterloo, 1999.

[KLW14] Philipp Kindermann, Fabian Lipp, and Alexander Wolff: Luatodonotes: Boundary labeling for annotations in texts. In Christian Duncan and Antonios Symvonis (editors): *Graph Drawing*, pages 76–88, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[Lit13] Grigori L. Litvinov: Idempotent and tropical mathematics; complexity of algorithms and interval analysis. *Computers & Mathematics with Applications*, 65(10):1483–1496, 2013.

[PC89] L. Paul Chew: Constrained Delaunay triangulations. *Algorithmica*, 1989.

[Pin98] Jean Eric Pin: Tropical Semirings. In J. Gunawardena (editor): *Idempotency (Bristol, 1994)*, Publ. Newton Inst. 11, pages 50–69. Cambridge Univ. Press, Cambridge, 1998.

[Str69] Volker Strassen: Gaussian Elimination is not Optimal. *Numerische Mathematik*, 1969.

[Wil12] Virginia Vassilevska Williams: Multiplying Matrices Faster than Coppersmith-Winograd. In *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '12, page 887–898, New York, NY, USA, 2012. Association for Computing Machinery.

Erklärung

Hiermit versichere ich die vorliegende Abschlussarbeit selbstständig verfasst zu haben, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, und die Arbeit bisher oder gleichzeitig keiner anderen Prüfungsbehörde unter Erlangung eines akademischen Grades vorgelegt zu haben.

Würzburg, den 30. October 2021

. . . . . .
Annika Förster