Karlsruhe Institute of Technology

Point Labeling with Leaders for Convex Boundaries

Diploma Thesis of

Neil Jami

At the Department of Informatics
Institute of Theoretical Informatics

Reviewers:	Prof. Dr. D. Wagner
	Prof. Dr. P. Sanders
Advisors:	Dr. Martin Nöllenburg
	Dipl.-Inform. Andreas Gemsa

Time Period: 01. November 2011 – 30. April 2012

Statement of Authorship

I hereby declare that this document has been composed by myself and describes my own work, unless otherwise acknowledged in the text.

Karlsruhe, 30th April 2012

Neil Jami

Abstract

This diploma thesis deals with the concept of convex boundary labeling. Given a set of points positioned in a map with a polygonal shape, the objective is to place, for each point, a rectangular label outside of the map, and connect it to the point with a leader. Generally, the labels are placed so that no two leaders intersect and so that the total leader length is minimized.

The main contribution of this work is the study of boundary labelings for maps with a convex polygonal shape. We consider axis aligned leaders with at most one bend, and assume that the labels can be placed anywhere to the right of the map. We study here three different models of the problem and present different algorithms to compute a crossing-free labeling with minimum leader length.

We describe the different algorithms and prove their correctness. The algorithms for the different models have a similar structure. They first compute a labeling with minimal leader length. For this purpose, each algorithm calls repetitively a second algorithm that computes a labeling with minimal leader length for a cluster of labels. In two of the three models, the second algorithm computes minimum weighted matchings. Since computing a matching takes a long time, we look for an alternative fast algorithm to avoid matching computations as much as possible. In a second step, the algorithms remove the remaining leader crossings in the computed labeling. Finally, we evaluate the quality and the performance of the implemented algorithms for practical inputs.

Deutsche Zusammenfassung

Diese Diplomarbeit beschäftigt sich mit dem Konzept von konvexer Randbeschriftungen. Gegeben eine Menge von Punkten in einer von einem Polygon begrenzten Karte, ist das Ziel, für jeden Punkt ein Label mit rechteckiger Form außerhalb des Polygons zu platzieren und durch einen Pfeil (Leader) mit dem Punkt zu verbinden. Somit werden Punkte in einer Karte annotiert. Üblicherweise werden die Labels so positioniert, dass die Leaders sich nicht kreuzen und die gesamte Leaderlänge minimiert wird.

Das Hauptthema in dieser Arbeit ist die Untersuchung von Algorithmen für konvexeRandbeschriftungen. Wir betrachten hier Leader, die aus höchstens zwei horizontalen oder vertikalen Segmente bestehen. Außerdem werden die Label rechts von der Karte platziert. Wir untersuchen hier drei Labeling Modelle und die entsprechenden Algorithmen, die kreuzungfreie Labelings mit minimaler Leaderlänge berechnen.

Wir beschreiben die verschiedenen Algorithmen und beweisen ihre Korrektheit. Die drei Algorithmen haben dieselbe Grundstruktur. Zunächst wird ein Labeling mit minimaler Leaderlänge berechnet, indem jeder Algorithmus iterativ einen weiteren Algorithmus aufruft, der ein Labeling eines einzigen Clusters mit minimaler Leaderlänge berechnet. Das Labeling eines Clusters wird in zwei der drei Modelle mithilfe von Matchings mit minimalem Gewicht berechnet. Da die Laufzeit von Matchingalgorithmen ziemlich hoch ist, wird eine schnellere Lösung benutzt, um so selten wie möglich einen Matchingalgorithmus aufzurufen. In einem zweiten Schritt entfernt jeder Algorithmus die eventuell vorhandenen Kreuzungen von dem Labeling minimaler Leaderlänge. Schließlich werden die Algorithmen implementiert, um ihre Laufzeit und Qualität für praktische Instanzen zu evaluieren.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Problem Definition . . . 2
- 1.3 Thesis Outline . . . 4
- 1.4 Related Work . . . 5

2 Preliminaries **7**
- 2.1 Notations . . . 7
- 2.2 Integer Linear Programming . . . 9
- 2.3 Matching Algorithms and Fixed-Labels Labeling . . . 10
- 2.4 Algorithms for Vertical Boundary . . . 11

3 Weakly-Aligned Boundary Labeling **15**
- 3.1 Labeling on a Single Edge . . . 16
  - 3.1.1 Leader Length Functions . . . 17
  - 3.1.2 Minimal Cluster Labeling using a Matching Algorithm . . . 18
  - 3.1.3 Minimal Cluster Labeling using a Sweep-line Algorithm . . . 19
    - 3.1.3.1 Position of a Cluster with Fixed Order . . . 20
    - 3.1.3.2 Finding an Optimal Order . . . 25
  - 3.1.4 Removing the crossings . . . 27
  - 3.1.5 Computing an Optimal Labeling . . . 30
  - 3.1.6 Linear Programming . . . 33
  - 3.1.7 Conclusion . . . 34
- 3.2 Convex Boundary with Big Slopes . . . 35
  - 3.2.1 Leader Length Functions . . . 36
  - 3.2.2 Minimal Cluster Labeling using a Matching Algorithm . . . 38
  - 3.2.3 Minimal Cluster Labeling using a Sweep-line Algorithm . . . 39
  - 3.2.4 Removing the crossings . . . 42
  - 3.2.5 Computing an Optimal Labeling . . . 43
  - 3.2.6 Integer Linear Programming . . . 45
  - 3.2.7 Conclusion . . . 46

4 Alternative Labeling Models **47**
- 4.1 Strictly-Aligned Boundary Labeling . . . 47
  - 4.1.1 General Results . . . 47
  - 4.1.2 Minimal Cluster Labeling using a Matching Algorithm . . . 50
  - 4.1.3 Minimal Cluster Labeling using a Sweep-line Algorithm . . . 51
  - 4.1.4 Crossing Removal . . . 51
  - 4.1.5 Conclusion . . . 55
- 4.2 Discrete Labeling using ILP. . . . 55
  - 4.2.1 Integer Linear Program . . . 56
  - 4.2.2 Choice of Slots . . . 57

4.2.3 Conclusion 58
4.3 Rectangular Block Labeling 58
4.3.1 Border With a Single Edge 63
4.3.2 Border With a Several Edges with Big Slopes 63
4.3.3 Conclusion 66

5 Evaluation **69**
5.1 Description of the Labeling Programs 69
5.2 Quality of the Labelings 71
5.3 Run Time Analysis 76

6 Conclusion **83**

Bibliography **85**

1. Introduction

1.1 Motivation

Over the last decades, the process of information visualization has become an important domain of research. Indeed, maps and illustrations represent an important visual component to communicate information. An easy to read map usually makes an explanation much easier than a whole written text. A particular interest has been given to the placement of extra information describing the maps. The reason is simple: manually generating a legend and describing an map takes a significant amount of the time for its creation.

This extra information usually takes the form of textual labels. The concept of linking each of these labels with some features in an map, namely called *labeling*, often represents the main contribution of the map. The map describes an environment, and these labels tell us what we are supposed to see, or where the element corresponding to each label is on the map. Applications exist in numerous and diverse areas such as cartography, anatomy, engineering and statistics.

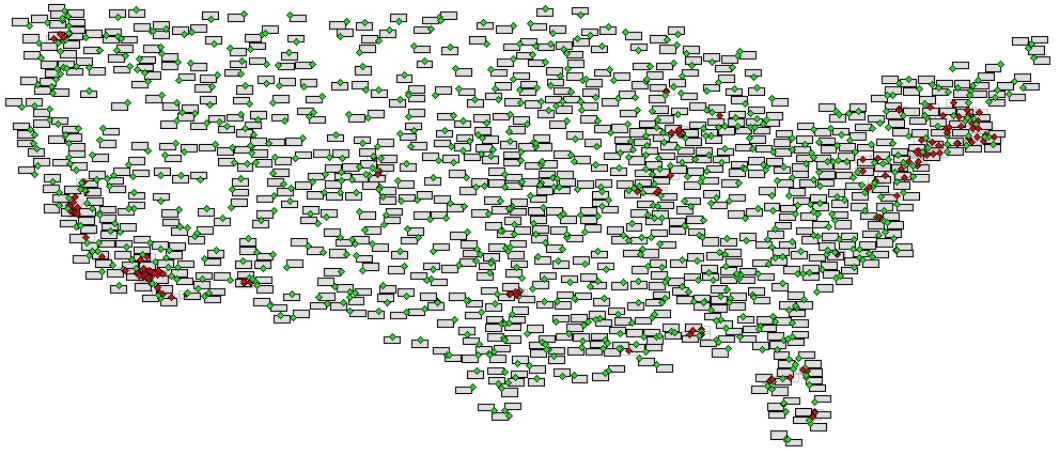


Figure 1.1: Example of labeling.

Cartographers have developed several criteria and ways of modeling the problem to compute high-quality labelings [Imh75, Yoe72]. Unfortunately, the majority of labeling problems are known to be NP-complete [Wag93]. Several approximation algorithms and heuristics have been suggested. the problem of map labeling has been widely researched on the

domains of cartography and computational geometry. The ACM Computational Geometry Impact Task Force report [Cca99] also denoted label placement as an important research area for the future years.

Until a few years ago, algorithms have been studied that place the labels on the map, next to the features they describe. However, such a placement is not always possible. For example, the problem of labeling is difficult in small areas with many features. Moreover, it is difficult to find the right size for the labels. Indeed, too small labels are difficult to read, whereas big labels hide parts of the maps, and thus disturb the readability of the map. In 2006, the concept of *boundary labeling* appeared in the algorithm literature [BKSW07]. Instead of placing the labels inside the map next to the features, the labels are placed outside, next to the border of the map. Each feature has then to be connected to a label with a curve. This new modeling has the double advantage to make dense regions easy to label and allow any size for the labels. Nevertheless, an important optimization problem is to compute curves between labels and feature that are easy for the human eye to follow.

Several simple but effective criteria have been suggested to produce clear boundary labelings, such as placing each label close to the corresponding feature.

In this thesis, we study a new variant of the boundary labeling problem. Most criteria have already be decided, but for most models the research only considered rectangular map borders. However, not all maps have a rectangular shape, and using a rectangular border may leave undesired unused space between the map and the labels. We consider here the possibility of a map delimited by a convex border.

1.2 Problem Definition

In our context, we define a map as a set of n points $\{P_0, \ldots, P_{n-1}\}$ inside a convex polygon R. Each point corresponds to a feature to annotate with a textual *label*, modeled by a rectangle with a fixed height. We define n labels $\{L_0, \ldots, L_{n-1}\}$ to connect to the points.We call *leader* the curve connecting a point to a label. We denote by *anchor point* or *anchor* the endpoint of a leader on the rectangular contour of a label. A placement of the labels and a drawing of the leaders is called *boundary labeling*. From now on, we will call *labeling* a boundary labeling.

In this thesis, we require the anchor of each leader to be the middle point of the left side of a label. Let r_{top} and r_{bottom} be the topmost and the bottommost points of the convex polygon R. We define the *right side* of the polygon R and the polygonal curve of R between R_{top} and r_{bottom} containing the rightmost point of R. In this thesis, the labels must be placed entirely outside of the map, and either touches the right side of the border or are farther away to the right of it. Moreover, The labels must not share the same y-coordinate, i.e., every horizontal line must not intersect two different labels. We denote by *right-labeling* a labeling imposing these constraints to the label position. Moreover, the leader will be polygonal curves composed of a vertical segment parallel to the left side of the labels, and a horizontal segment orthogonal to the left side of the labels. Such a leader is called *po-leaders*. Figure 1.2 shows an example of right-labeling using po-leaders.

In the literature, further shapes of leader have also been defined, like direct-leaders, opo-leaders [BKSW07], and do-leaders [BHKN09] as presented in Figure 1.3.

In order to make the labeling as easy to read as possible, we add constraints that our labeling must respect. One of the most used constraint is to minimize the number of crossings between the leaders and between the leaders and the labels. We call then *leader crossing* an intersection between two leaders and *leader-label crossing* an intersection between a

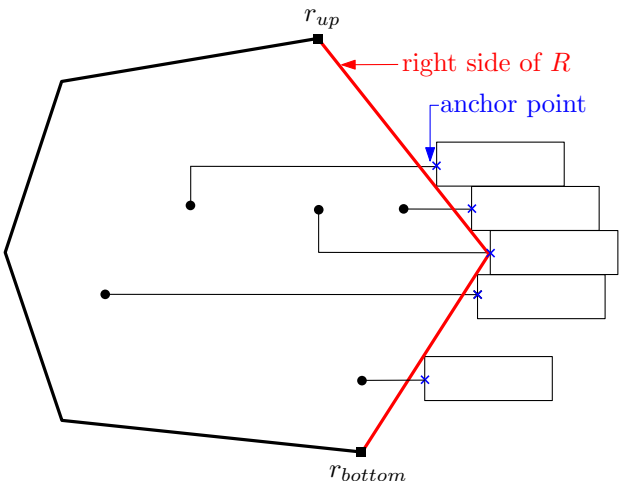
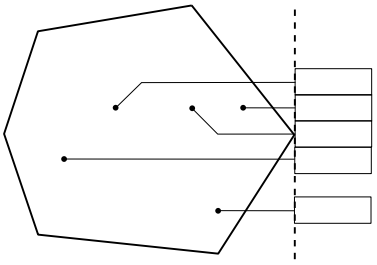
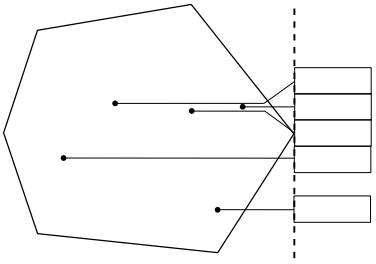


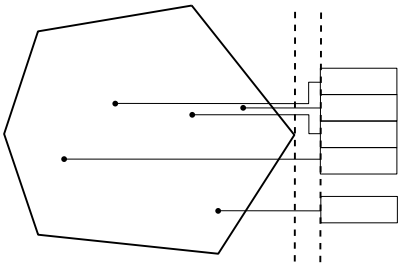
Figure 1.2: Example of right-labeling using po-leaders.



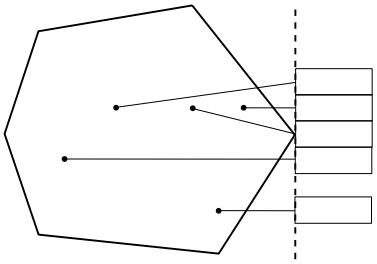
(a) do-leader: a diagonal segment and an orthogonal segment



(b) do-leader: an orthogonal segment and a diagonal segment



(c) opo-leader: an orthogonal segment, a parallel segment and an orthogonal segment



(d) direct-leader: a single direct segment

Figure 1.3: Some other leader types, studied when the right side of the border is vertical.

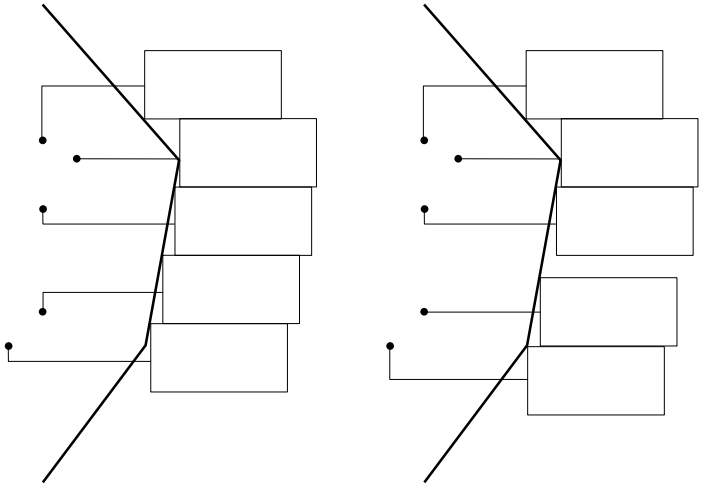
label and a leader. We say that a labeling is *crossing-free* when it contains no crossing between two leaders or between a leader and a label. As we will see, given a set of points and a border R, we can always find a crossing-free right-labeling using po-leaders.

We consider in addition a badness function, associating a labeling to a badness value. The minimum of this function must provide a clear labeling. Intuitively, a labeling seems to be easier to read when the labels are close to the point they are connected to. Another possibility would be to minimize the number of bends of the labels.

Further conditions for the position of the labels may be added. We define a *labeling with fixed labels* as a labeling so that a number m *slots* of possible label positions are fixed, and we have to decide which label will be placed in which slot and how to connect these labels to the points. On the other side, we denote by *labeling with sliding labels* a labeling where

the labels can have any position, and we add a constraint of non-intersection of the labels.

Furthermore, we define a concept of cluster-labeling: a *cluster* of labels is a block of several labels so that the top-side of each label coincides with the bottom-side of the label above it in the cluster. We denote by *cluster-labeling* a labeling so that every labels has to be in the same cluster. A labeling composed of several clusters is called *clustered-labeling*, as illustrated in Figure 1.4.



(a) Cluster labeling: a single cluster

(b) Clustered labeling with two clusters

Figure 1.4: Illustration of cluster of labels

A labeling is composed of one or several clusters of labels connected to the points. When annotating features, many people tend to align vertically the texts which are close to each other, in order to make them look more uniform. Therefore, it may be suggested to give a rectangular shape to the clusters by aligning vertically the left side of the labels in each cluster.

In this thesis, we compute crossing-free right-labelings using po-leader with a minimal total leader length. The function to minimize is the sum of the L^1 distances between each point and the label it is connected to. This badness function makes computations easy and seems to be a good model to put the labels as close as possible to the points they are connected to. We denote by *minimal labeling* a labeling that minimizes the total leader length and by *optimal labeling* a crossing-free minimal labeling.

1.3 Thesis Outline

This work is structured as follows: In the next section, we present an overview of the related work. In Chapter 2, we introduce specific notations that we will use in the thesis, and present some results from existing work that we will use and generalize, such as the Integer Linear Programming, matching Algorithms and an algorithm to compute the desired labeling when the right side of the polygon R is a vertical segment. In Chapter 3 and Chapter 4, we study the labeling problems for different constraints on the labels. Chapter 3 presents the main results for the so-called *weakly-aligned boundary labelings*, where the labels can be placed anywhere to the right of the map. In Chapter 4, we consider two variants of this labeling. In a *strongly-aligned boundary labelings*, labels are required to touch the boundary, and in a *rectangular-block labeling*, the clusters have a rectangular shape, and the left side of the labels in a same cluster are vertically aligned. Chapter 5 provides experimental results on the quality and the time-complexity of the computed labeling. Chapter 6 concludes this thesis.

1.4 Related Work

The concept of boundary labeling has been introduced in 2006 by Bekos et al. [BKSW07]. In this paper, the authors considered several models for rectangular maps and fixed label positions. The labels are fixed on one, two or four sides of the map. Three kinds of leaders were introduced, that is the direct leaders, the one-bend po-leaders and the two-bend opo-leaders. These three kinds of leader are presented in Figure 1.3. and their name describe their shape. An opo-leader is composed of an 'o' segment, i.e., orthogonal to the the left side of a label, then of a 'p' segment, i.e. parallel to the the left side of a label, and of a second 'o' segment. The authors studied the labeling problem with fixed ports and with ports sliding on the labels, where the *port* of a leader is its endpoint touching the label. Two different optimization functions have been presented, namely the minimization of the total leader length, and the minimization of the number of *bends* of the leaders. In particular, a algorithm has been developed to compute in $O(n^2)$ a crossing-free labeling with po-leader minimizing the total leader length with either fixed port or with sliding ports. Further algorithms have been developed for direct-leaders and for opo-leaders.

In another paper, Bekos et al. [BKPS06] studied boundary labelings with opo-leaders for multiple stacks of labels. The labels are then placed on several columns on one side of a rectangular boundary.

In order to overcome the abrupt form of the bend in po-leaders, Benkert et al. [BHKN09] studied the boundary labeling with a new type of leader called *do-leader*, where 'd' stands for 'diagonal segment'. Unfortunately, boundary labelings with do-leaders may not always be *feasible*. For example, if the points are concentrated in the same area next to the border, the diagonal part may not allow us to place a label far enough from the point. For one-sided labelings using po-leaders, the authors developed an algorithm to compute a crossing-free labeling with fixed label positions and minimal leader length in $O(n \log n)$ time. The concept of do-leader has been generalized to *octilinear leaders* [BKNS09], where new leader types pd- and od-leaders have been added. Then, using two types of leaders, for example do- and od-leaders, the problem of infeasible labelings has been solved. The authors developed algorithms to compute in (n^3) time crossing-free labelings with minimal leader length for each type of octilinear leaders.

In 2008, Bekos et al. [BKPS08] considered the model where features are not points sites but area features, i.e., a region of the map. They allowed the leader to be attached to any position in the area feature. The authors generated labelings with a min-cost bipartite matching algorithm, with a time complexity of $O(n^3)$.

Lin et al. [LK08] studied the possibility to connect a label to several points Such labelings are particularly interesting when several point on the map represent the same object type. Reducing the number of labels avoids redundancy of text and allows to use bigger labels with more text. They proved the NP-completeness of this problem and presented approximation algorithms.

In 2010, Čmolík et al. [vB10] employed fuzzy logic and greedy optimization to create layout aware labelings. The authors modeled features as non intersecting surfaces, and each label has to be connected to a dynamic point in such a surface. In order to improve the quality of the labeling, the border of the map has been taken into account to compute a reasonable input position of the labels. However, the algorithm only compute an approximation of a labeling with minimal leader length and may not be suited for dense maps.

Nöllenburg et al. [NPS10] envisaged the visualization boundary labelings for dynamic maps, where the user can interactively zoom in or out. Therefore, the view of the map is not fixed, and the number and the position of points and labels continuously change.

This labeling has been called *dynamic boundary labeling*, and has been mostly studied for one-sided or two-sided labeling in rectangular maps.

Finally, Gemsa et al. [GHN11] studied boundary labelings for panorama images, where points in a rectangle R are connected to disjoint rectangular labels places above R in k rows. Each points in connected to its label by a vertical leader that does not intersect any other label. The authors presented polynomial time algorithms that either minimize the number of rows to place the labels, or maximize the number of labels that can be placed in a fixed number of k rows. For weighted labels, the problem is shown to be NP-hard.

An overview of the main results regarding the boundary labeling problem has been written by Kaufmann [Kau09], and an extensive bibliography is maintained by Wolff [Wol96].

2. Preliminaries

In this chapter, we present our notations and the existent algorithms that will help us to compute our convex-boundary labelings. We first present the notations used in this thesis. Then we will see what is an integer linear program. In the next section we define the matching problems and give two matching algorithms to solve the problem of optimal labeling for fixed label positions. We finally describe the structure of the algorithm proposed by Nöllenburg et al. [NPS10] to compute an optimal labeling when the right side of the boundary is vertical.

2.1 Notations

The points are denoted by $\{P_i = (x_i, y_i) \mid i = 0, \ldots, n-1\}$, and the labels by $\{L_j = (t_j, s_j) \mid j = 0, \ldots, n-1\}$. Each label is supposed to have a rectangular form, and is then delimited by four *corners*. The coordinates of a label corresponds to the coordinates of its anchor, i.e., the vertical midpoint of its left side. We suppose that both the set of points and the set of labels are sorted by increasing y-coordinate, namely $y_0 < y_1 < \ldots < y_{n-1}$ and $s_0 < s_1 < \ldots < s_{n-1}$. We denote by $\ell(P_i, L_j)$ the leader connecting the point P_i to the label L_j. The length of the leader $\ell(P_i, L_j)$ is given by the L^1 distance between P_i and L_j:

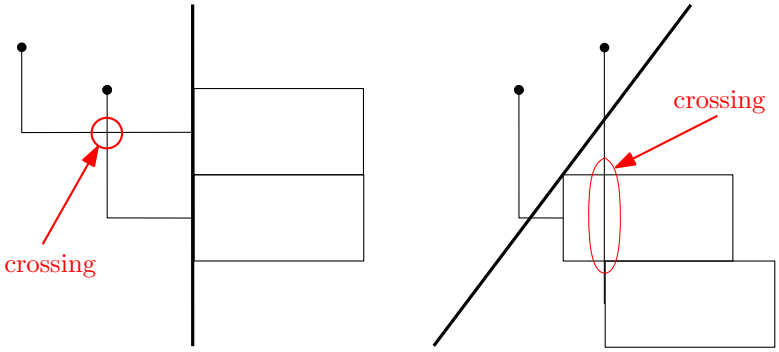
$$|\ell(P_i, L_j)| = |y_i - s_j| + |x_i - t_j|$$

We define the *vertical length (horizontal length)* of a leader $\ell(P_i, L_j)$ as the the distances of y-coordinates (x-coordinates) between P_i and L_j. We denote by $\mathcal{L}$ a boundary labeling, and introduce a bijective function σ from the point indices to the label indices so that in the labeling $\mathcal{L}$ the point P_i is connected to the label $L_{\sigma(i)}$. The function σ is called *order of labels*. We denote by *Y-order* the order of labels so that the i-th bottommost label L_i is connected to the i-th bottommost point P_i for each index i.

We call *badness* of a labeling $\mathcal{L}$ its total leader length in this labeling, and we denote by bad this function:

$$\text{bad}(\mathcal{L}) = \sum_{i=0}^{n-1} |\ell(P_i, L_{\sigma(i)})|$$

A labeling is called *minimal* when it has a minimum badness. In this thesis, we denote by *leader length function* a function computing the length of a leader given the position of its point and label. Moreover, we call *leader crossing* an intersection between two leaders, and *leader-label crossing* an intersection between a label and a leader. Both of these crossings



(a) Leader crossing (b) Leader-label crossing

Figure 2.1: Illustration of leader crossings and leader-label-crossings

are illustrated in Figure 2.1. A *crossing-free* labeling contains no leader crossing and no label crossing. Our objective is to compute an *optimal labeling*, that is, a minimal and crossing-free labeling.

Further notations are defined for the convex boundary. The map is delimited by a convex polygon R and contains n points. We define the *border* as the right side of polygon R delimited by its topmost and its bottommost points. The border is composed of k segments called *edges*. The edges are denoted by $E_1, \ldots, E_k$ sorted from the bottom to the top. We denote by *nodes* the endpoints $N_0, \ldots, N_k$ of the edges, so that each edge E_i is delimited by the nodes N_{i-1} and N_i. Since the labels are placed right from the border, this border is characterized by an equation E giving the x-coordinate of a label from its y-coordinate:

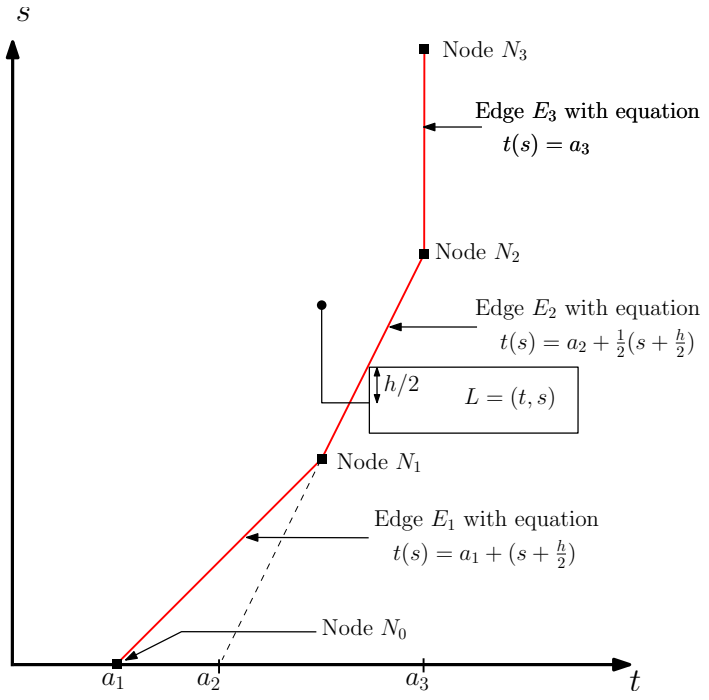
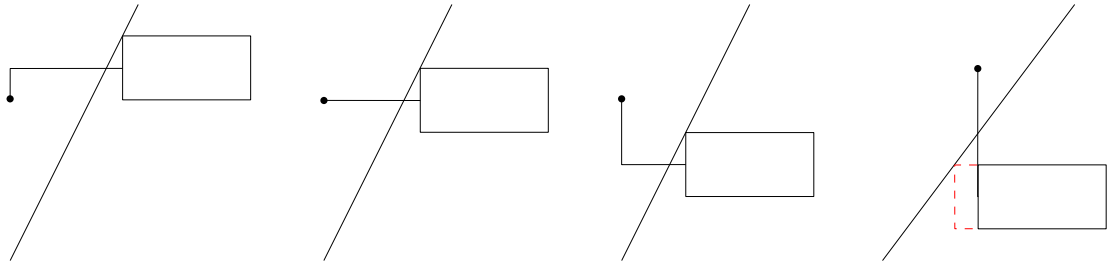


Figure 2.2: Label $L = (t, s)$ touching a border E with three edges E_1, E_2 and E_3.

Since we look for a right-sided labeling with minimal length, we only consider the position (t_j, s_j) for a label so that there is no valid position (t, s_j) inducing a smaller leader length. Consider the leftmost possible position (t', s_j) for a label when the vertical midpoint is

s_j, i.e., the label touches the border with its left side. Let $P_i : (x_i, y_i)$ be the point the label is connected to. If $t' > x_i$, then $t_j = t'$ is the optimal position for the label at the y-coordinate s_j, and any other position t increases the leader length by $t - t'$. In this case, we say that the label is *on the border*. Otherwise, the optimal position for the label has x-coordinate $t_j = x_i$, and any other label position with x-coordinate $t \neq x_i$ and y-coordinate s_j increases the leader length by $|t - x_i|$. In the last case the label is said to be *shifted*.

In addition, we define three types of leaders. We call a leader *upward* when the incident point is below the incident label, *downward* when the incident point is above the incident label, and *straight* otherwise.



(a) upward (b) straight (c) downward (d) downward shifted

Figure 2.3: Different Types of Leaders.

Sometimes, several labels $L_i, L_{i+1}, \ldots, L_j$ will be regrouped so that the bottom side of each label L_k coincides with the top side of the label L_{k-1} and its top side coincides with the bottom side of the label L_{k-1}. Such a group of labels is called *cluster of labels*. We note that the position of a cluster is entirely described by the position of its bottommost label. We define a *cluster labeling* as a position of the cluster and a bijection σ from the points connected to the cluster to the labels. A cluster labeling is called minimal (resp. optimal) when no other labeling of this cluster has a lower badness (resp. when it is minimal and crossing-free).

We define the operation *moving a cluster upward (downward)* by a distance of ε as increasing (decreasing) the y-coordinate of each label of the cluster by ε and then computing their new optimal x-coordinate. Moreover, we define the operation of *switching two labels* L_i and L_j as switching the values of their y-coordinate, and then recomputing the optimal x-coordinate associated to this position. Given a labeling, we call the *gain* G of a modification the difference of badness after and before this modification. If the gain G is negative, then the new labeling has a badness smaller than the initial labeling. A modification can be moving labels or switching labels. The badness function is minimal if no modification has a negative gain.

2.2 Integer Linear Programming

A *linear program* is defined by a linear function to optimize and a set of linear constraints, which are equalities or inequalities, to respect. Every linear program can be reduced to the following *canonical* form:

$$\begin{aligned} &\textit{minimise } c^T x \\ &\textit{subject to } Ax \leq b \\ &\qquad\qquad x \geq 0, \end{aligned}$$

where $c \in \mathbb{R}^n$ is a n-dimensional vector defining the objective function, $x \in \mathbb{R}^n$ is the n-dimensional vector of the variables, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ are a matrix and a vector

defining the constraints of the problem. A linear program can be solved in polynomial time, using for instance Karmakar's interior point method [Kar84].

In case of the variables and constraints having integer values, this problem is called *Integer Linear Program*. Many optimization problems can be modeled as Integer Linear Programs, but not as Linear Programs, which makes Integer Linear Programming a powerful tool to use in optimization. Unfortunately, this problem is known to be NP-hard [GJ79]. Chandru and Rao gave a survey [CR99] containing a good presentation of Integer Linear Programming and the theory of discrete optimization in general. There exist several solvers such as *gurobi*[1] or *lp_solve*[2] to solve integer linear programs, using heuristics to compute quickly a first solution close to the optimum, then slowly improving this solution.

2.3 Matching Algorithms and Fixed-Labels Labeling

The theory of graphs is an important domain in applied mathematics and algorithmics. A *graph*[3] is a tuple $G = (V, E)$ composed of a set V of $|V|$ *vertices* and a set E of $|E|$ *edges* $\{u, v\} \in V \times V$. Given an edge $e = \{u, v\}$, we say that the vertices u and v are *incident* to the edge.

One of the most well-known problems in graph theory is the matching problem. A *matching* is a set of edges so that no two edges are incident to the same vertex. The matching problem consists in searching a matching with maximum cardinality.

We call *bipartite matching problem* a matching problem in a bipartite graph, i.e., in a graph where the vertices are divided into two sets N_1 and N_2 so that each edge connects a vertex from N_1 to a vertex from N_2. A simple example is the assignment problem. We dispose of n_1 persons and n_2 tasks. We want to accomplish as many task as possible. We create then a graph with $n_1 + n_2$ vertices corresponding to the personnel and the tasks. We create an edge between the person P_i and the task T_j if P_i has the necessary competences to accomplish T_j. The assignment problem corresponds to the search for a maximum cardinality matching.

We define now a cost function over the edges:

$$\text{cost} : E \longrightarrow \mathbb{N}$$

We define a *maximum weighted matching* a matching $S^* = \{e_1, \ldots, e_k\}$ in a graph with a cost function to be a matching with maximum cost.

$$\text{cost}(S^*) = \sum_{e \in S^*} \text{cost}(e)$$
$$\text{cost}(S^*) \geq \text{cost}(S) \text{ for every matching S.}$$

We call *maximum weighted bipartite matching problem* a maximum weighted matching problem in a bipartite graph.

Let's go back to the labeling problem. Suppose that we have n points and $m \geq n$ fixed and non-intersecting slots for the labels. The objective is to compute a minimal labeling with fixed labels slots.

As we saw in section 2.1, for one-sided boundary labeling, the border is described by an equation giving the x-coordinate of a label from its y-coordinate. Therefore, we can suppose that only the y-coordinates of the labels are fixed. Then, for each pair (P_i, L_j) of

[1] see http://www.gurobi.com/products/gurobi-optimizer/gurobi-overview
[2] see http://lpsolve.sourceforge.net
[3] We consider here only the notion of undirected graphs.

point and label, we can compute the length of the leader $\ell(P_i, L_j)$ if P_i is connected to L_j. Therefore, computing a minimal labeling is equivalent to computing a minimum weighted bipartite matching. Our problem is thus solved by the search of a maximal weighted matching in the following bipartite graph $G = (V_1 \cup V_2, E)$ with:

- V_1: set of n points $P_0, \ldots, P_{n-1}$.
- V_2: set of $m \geq n$ labels $L_0, \ldots, L_{n-1}$.
- $E = V_1 \times V_2$, each edge $\{P_i, L_j\}$ has cost $C - |\ell(P_i, L_j)|$, where C is a constant greater than any leader length:

$$C > \max_{leader\ \ell} |\ell|$$

Many algorithms exists for the matching problem. E. L. Lawler [Law76] presented several polynomial time algorithms to compute any kind of matchings.

For bipartite graphs, the Hungarian algorithm [Kuh55] provides an efficient solution in $O(|V| \cdot |E|)$ time. The following theorem provides thus a first algorithm for *Fixed-Label labeling*, that is labeling so that the y-coordinates of all labels are fixed.

Theorem 1. *Given a set of n points and m non intersecting sites for the labels. The Hungarian Algorithm [Kuh55] computes in $O\big((n+m) \cdot n \cdot m\big)$ time a minimal labeling.*

Furthermore, Vaidya [Vai89] studied graphs representing cartographic data, so that the cost function over the edges is a distance function under some metric. In particular, he studied *perfect matching* in *complete graphs*: a matching is said to be *perfect* when every vertex is incident to one of the edges in the matching. To create perfect matchings, the two sets of vertices N_1 and N_2 must then have the same size. A graph is called *complete* when there is an edge connecting every two vertices. A bipartite graph is complete when there is an edge connecting every vertex of the set N_1 to every vertex of the set N_2.

With the help of geometry data, he created an algorithm with $O\big(|V|^2 \cdot (\log |V|)^3\big)$ time complexity to compute a minimum weighted perfect matching in a complete bipartite graph, when the *cost* function follows the L^1 or the L^∞ metric. For po-leaders, the length computed is the L^1 distance. We obtain the following theorem:

Theorem 2. *Given a set of n points and n possible sites for the labels, the Algorithm of Vaidya [Vai89] computes in $O\big(n^2 \cdot (\log n)^3\big)$ time a minimal labeling.*

2.4 Algorithms for Vertical Boundary

This section presents the basics of our new models and algorithms. We suppose here that the right border is a vertical line. We compute clustered-labelings with sliding labels, i.e., labelings which may contain several clusters of labels and where the possible positions of the labels are not fixed. The objective is to compute an optimal labeling, that is a crossing-free labeling with minimal leader length. This problem has already been studied by Nöllenburg et al. [NPS10], for one-sided labeling in a rectangular boundary and with sliding labels. The objective of this thesis is to generalize this model to the case of a convex boundary. We summary here the main results, which will help us developing new algorithms.

Recall that σ denotes the order of the labels, i.e., in a labeling $\mathcal{L}$ the point P_i is connected to the label $L_{\sigma(i)}$. For a one-sided labeling using po-leaders and with a vertical border, all

labels have the same abscissa t, and it is sufficient to minimize the vertical leader length in a labeling $\mathcal{L}$.

$$bad(\mathcal{L}) = \sum_{i=0}^{n-1} \big(|s_{\sigma(i)} - y_i| + |t - x_i|\big) = \sum_{i=0}^{n-1} \big(|s_{\sigma(i)} - y_i|\big) + C,$$

where C is a constant. We will see at the end of this part an algorithm with $O(n^2)$ time complexity to remove crossings in a minimal labeling. Thus, we will now look for a labeling with minimal badness, and then use the other algorithm to remove the crossings.

Given a single label, positioning the label at the same y-coordinate as the point minimizes the leader length. This leader is then straight. The farther the label is moved away from this position, the greater the length of the leader gets. This property is formalized by the following lemma:

Lemma 1. *Let $f_i(s_j)$ be the function giving the length of the leader $\ell(P_i, L_j)$ from the y-coordinate s_j of a label L_j. The function $f_i(s_j)$ is convex and has a unique minimum for $s_j = y_i$.*

When the labeling consists of several labels whose optimal positions intersect, these labels will be regrouped into clusters. An optimal position is computed for each cluster of labels.

For each cluster, we must find out the position of the cluster and the order σ of the labels that minimize the total leader length. The following lemma tells that there exists a labeling with minimal leader length connecting the label L_i to the point P_i for each i.

Lemma 2 (corresponds to theorem 6 in [BKSW07])**.** *There exists a minimal labeling using the Y-order.*

We now assume that the i bottommost label L_i is connected to the i bottommost point P_i, i.e., $\sigma(i) = i$, for each i. The following lemma gives the optimal position of a cluster of n labels from the coordinates of the points $P_0, \ldots, P_{n-1}$:

Lemma 3 (Lemma 3.2 in [NPS10])**.** *A position of the cluster with at most $\frac{n}{2}$ upward leaders and at most $\frac{n}{2}$ downward leaders is optimal. In particular, this condition is satisfied when the leader corresponding to the median of the set $\{y_i - i \cdot h\}$ is straight, where h is the height of the labels.*

Nöllenburg et al. (Lemma 4.1 in [NPS10]) proposed a data structure which can be build in $O(n \log n)$ time and that allows to do median queries for clusters in $O(\log n)$ time. Lemma 3 induces an algorithm giving the optimal position of a cluster in $O(\log n)$ time given this data structure, if we do not compute the exact position of each label. Algorithm 1 computes a grouping of the labels into clusters inducing a minimal labeling. We will see in Section 3.1 that it is due to the convexity of the leader length functions stated in Lemma 1.

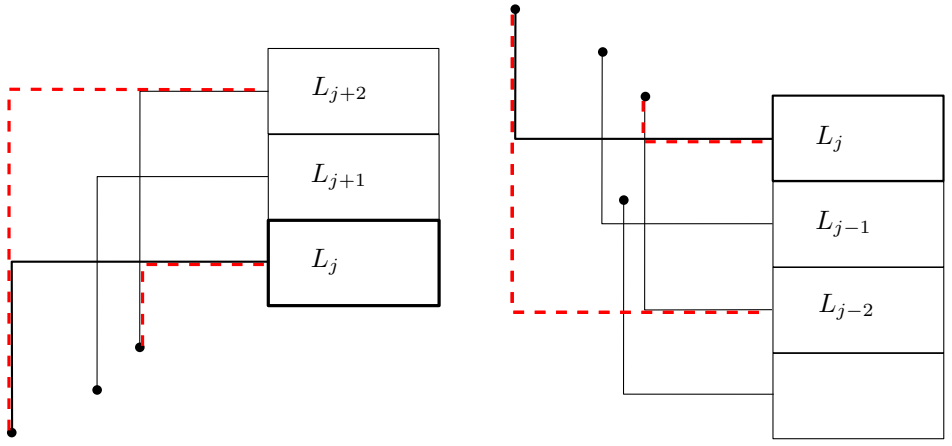
By combining it with the Upward-downward algorithm with Lemma 3 and the data structure proposed by Nöllenburg et al., we create an algorithm computing a minimal labeling in $O(n \log n)$ time. Finally, we have to remove leader crossings in the computed minimal labeling. The following lemma gives an interesting property about the leader crossings in a minimal labeling.

Lemma 4 (Lemma 1 in [BHKN09])**.** *In a minimal labeling, there are no crossings between an upward leaders and a downward leader. Moreover, no straight leader can be involved in a crossing with an upward leader and a crossing with a downward leader at the same time.*

Algorithm 1: Upward-Downward Algorithm

Data: $\{P_0, \ldots, P_{n-1}\}$.
Result: minimal labeling.

Sort the points by increasing y-coordinate;
for $i = 0$ *to* $n - 1$ **do**
    // add P_i in the labeling;
    Position L_i at its optimal position;
    Let C_i be the cluster containing L_i;
    while C_i *intersects a cluster* C_j **do**
        Fusion C_i and C_j;
        Compute a minimal cluster labeling for C_i;
return the created labeling;



(a) L_j is connected by a downward leader (b) L_j is connected ny an upward leader

Figure 2.4: Removing the crossing of the leader connected to the label L_j. The dashed lines in red represent the new connections between points and leader. We note in Figure (b) that changing the connection of two downward leader can create a new crossing, with a downward leader connected to a label below L_j.

We denote by *downward crossing* a crossing involving a downward leader, and by *upward crossing* a crossing involving an upward leader. See Figure 2.4 for an illustration. We know from Lemma 4 that the labels involved in crossing can be subdivided into two disjoint sets, the first one containing the upward crossings and the second one containing the downward crossings. Moreover, when we switch two label positions to remove a downward crossing, we do not create any upward leader, and vice-versa. Therefore, these two sets of crossing leaders can be made crossing-free independently.

When we switch the position of two labels L_i and L_j corresponding to an upward crossing, we can not create a crossing involving a leader incident to a label below L_i and L_j. Likewise, when we switch the position of two labels L_i and L_j corresponding to a downward crossing, we can not create a crossing involving a leader incident to a label above L_i and L_j. From these observations, Algorithm 2 has been developed to remove leader crossings. Figure 2.4

shows what happens in the algorithm.

Algorithm 2: Crossing-Removal Algorithm

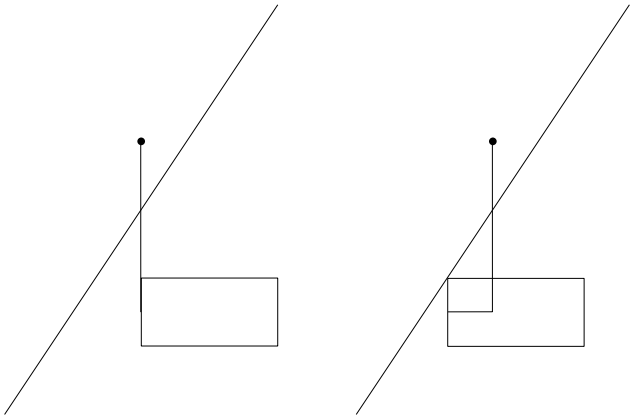
Data: minimal labeling $\mathcal{L}$.
Result: remove the crossings of $\mathcal{L}$ without increasing the badness.
for $j = n-1$ *downto* 0 **do**
  // the leaders incident to labels $L_j,\ldots, L_{n-1}$;
  // are not involved in downward crossings ;
  if $\ell(P_{\sigma^{-1}(j)}, L_j)$ *involved in a downward crossing* **then**
    $\ell(P_{\sigma^{-1}(k)}, L_k)$:= downward leader with the rightmost crossing with $\ell(P_{\sigma^{-1}(j)}, L_j)$.;
    Switch the position L_j and L_k.;
for $j = 0$ *to* $n-1$ **do**
  // the leaders incident to labels $L_0,\ldots, L_j$;
  // are not involved in upward crossings ;
  if $\ell(P_{\sigma^{-1}(j)}, L_j)$ *involved in an upward crossing* **then**
    $\ell(P_{\sigma^{-1}(k)}, L_k)$:= upward leader with the rightmost crossing with $\ell(P_{\sigma^{-1}(j)}, L_j)$.;
    Switch the position L_j and L_k.;
return the labeling;

We now state the final result for a vertical border:

Theorem 3. *For a one-sided labeling using po-leaders and with a vertical border, an optimal labeling can be computed in $O(n^2)$ time.*

3. Weakly-Aligned Boundary Labeling

In this chapter, we consider a first model of labeling for convex boundary. In Section 2.4, we considered right-sided po-labelings for vertical boundary. Given a y-coordinate of a label, the x-coordinate of the label that minimizes the leader length corresponds to a label touching the border. However, when the boundary is not vertical, the x-coordinate of the label that minimizes the leader length can be shifted to the right of the border, see Figure 3.1.



(a) A shifted label

(b) A label crossed by the leader it is connected by

Figure 3.1: Unclear position of a label

On the other side, to make a labeling clear, a leader must not cross the label it is connected to, as shown in Figure 3.1. Our first model consists of allowing labels to be *shifted* to the right of the border, so that no leader crosses the label it is connected to. For a given y-coordinate of a label, we will always position the label at the x-coordinate which minimizes the leader length. Therefore, no label will be placed to the right of the point it is connected to. We call this model of labeling *weakly-aligned boundary labeling*, or *WAB-labeling*, and every labeling in this chapter will refer to a WAB-labeling.

In a first section, we will assume that the border consists of a single edge. We will generalize the algorithms from Section 2.4 to the case of the edge of the border being not necessarily vertical, and develop an algorithm to compute an optimal labeling when the edge of the

border has a slope greater than 1. In a second section, we will study the case of a convex border composed of several edges with slopes greater than 1.

3.1 Labeling on a Single Edge

In this section, the border consists of a single edge positioned to the right of the set of points. Therefore, the labels are always placed further to the right of the border so that the leader length is minimal. To minimize the leader length, the label may have to be shifted. The objective here is to compute an optimal labeling, i.e., a a crossing-free labeling with minimal leader length. Figure 3.2 shows an example of labeling for this section.

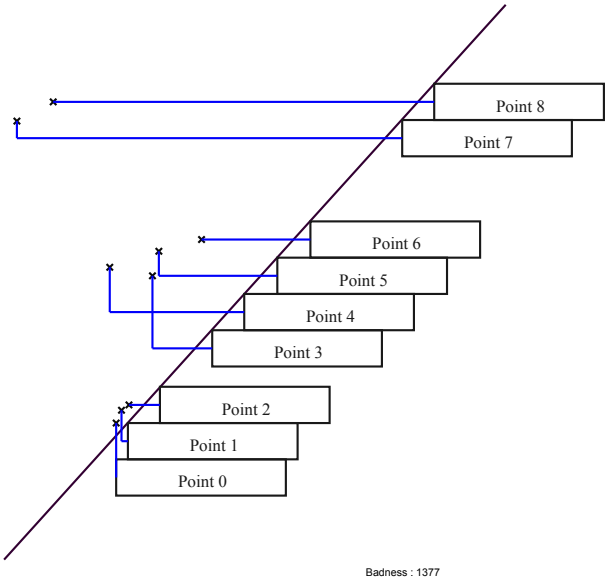


Figure 3.2: Example of optimal labeling looked for in this section. The bottommost label is shifted.

Since the case of an edge with a positive slope is symmetric to the case of an edge with a negative slope, we will assume here that the slope of the edge composing the border is positive. Then, each label connected by an upward leader cannot be left of the point it is connected to. Therefore, shifted labels can only be connected by downward leaders. We distinguish three kind of downward leaders:

1. *downward-straight* leaders: the label is not shifted, but the leader is vertical.
2. *downward-shifted* leaders: the label is shifted and the leader is vertical.
3. *downward-regular* leaders: the leader has a vertical and a horizontal segment.

In order to avoid confusions, we will call *horizontal-straight* leaders what we called straight leaders in the previous chapter.

We denote by *leader length function* a function computing the leader length of a label given its y-coordinate or a function computing the total leader length for a cluster of labels given the y-coordinate of it bottommost label.

In this section, we will first study the properties of the leader length functions. Then we will develop two algorithms to compute minimal cluster labelings, i.e., labeling with a minimal total leader length such that every label is in the same cluster. We will also study the crossing removal step before developing an algorithm that computes an optimal cluster labeling using the Upward-downward algorithm. We will conclude after describing a simple integer linear program to test experimentally the correctness of the written program for our algorithm.

3.1.1 Leader Length Functions

Let s_j be the y-coordinate of a label L_j. When the label touches the border, its x-coordinate t_j can be calculated with the border equation: $t_j(s_j) = \frac{s_j}{m} + a$, with $m > 0$. A vertical border corresponds to the case $m = \pm\infty$.

Consider a point P_i. We denote by $f_i(s_j)$ the length of the leader incident to P_i when the label it is connected to has y-coordinate s_j. The following lemma states an interesting property of these leader length functions when the value of the slope m is bigger than 1:

Lemma 5. *The function $f_i(s_j)$ corresponding to any point P_i is piecewise linear and convex. Moreover, the endpoints of the segments of $f_i(s_j)$ correspond to a horizontal-straight leader or to a downward-straight leader.*

Proof. Let $P_i = (x_i, y_i)$ be a given point of the map, connected to a label $L_j = (t_j, s_j)$. Let's compute the length of the leader depending on the y-coordinate s_j of the label. Figure 3.3 shows the global form of the function.

Let z_i be the y-coordinate of the label when its x-coordinate is x_i. Since the label cannot be moved left of the point, every position of the label below this one makes L_j a shifted label:

$$\forall s_j \leq z_i,\ |\ell(P_i, L_j)| = y_i - s_j$$

This length is greater than the length of the leader when $s_j = z_i$.

If the leader is upward, then the length of the leader depending on $s_j > y_i$ is:

$$|\ell(P_i, L_j)| = (s_j - y_i) + (t_j - x_i)$$

Since the label is on the border, we have $t_j = a + \frac{s_j}{m}$, and thus:

$$\forall s_j > y_i,\ |\ell(P_i, L_j)| = s_j \cdot (1 + \frac{1}{m}) + a - x_i - y_i$$

This length is greater than the length of the leader when $s = y_i$, i.e., when the leader is horizontal-straight.

Finally, if the leader is downward but not shifted, then its length is:

$$|\ell(P_i, L)| = (y_i - s_j) + (t_j - x_i) \text{ with } t_j = a + \frac{s_j}{m}$$

Thus:

$$\forall s \in [z_i, y_i],\ |\ell(P_i, L_j)| = s_j \cdot (\frac{1}{m} - 1) + a - x_i + y_i$$

Therefore, the optimal position of a label depends on the value of the slope m: If the slope is *big*, i.e., $m \geq 1$. then the minimal leader length corresponds to a *horizontal-straight* leader, when the label has y-coordinate $s_j = y_i$. However, when the slope is *small*, i.e., $m \leq 1$, the minimal leader length corresponds to a *downward-straight* leader, when the label has y-coordinate z_i.

Therefore, the leader length function f_i is piecewise linear and composed of three segments with respective slope -1, $\frac{1}{m} - 1$, and $\frac{1}{m} + 1$ when sorted by increasing s values. Since we supposed $m > 0$, the second slope $\frac{1}{m} - 1$ is greater than -1 and lower than $\frac{1}{m} + 1$. Thus, the leader length function f_i is convex. □

We deduce from this lemma the following corollary, to compute a minimal labeling for a cluster of labels.

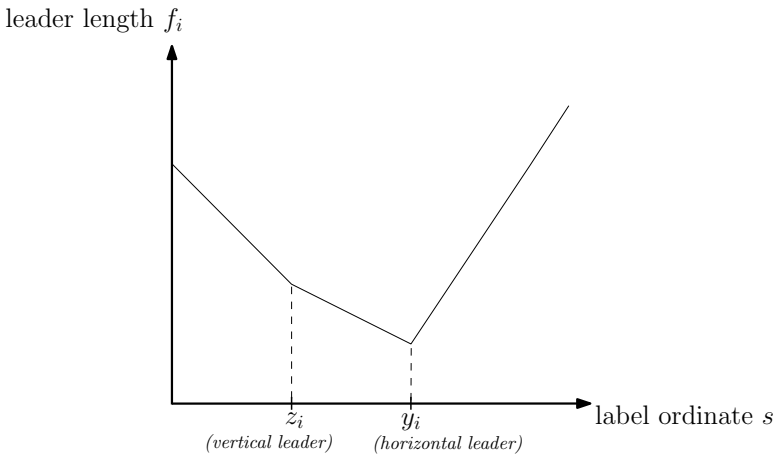


Figure 3.3: Length of a leader as a function of the y-coordinate s of its label, for a single-edge border with big slope: $m > 1$

Corollary 1. *Given a set of labels constituting a cluster. There exists a minimal cluster labeling with a horizontal-straight leader or a downward-straight leader.*

Proof. Consider a cluster C composed of n labels $L_0, \ldots, L_{n-1}$ sorted by increasing y-coordinates. Each of these labels has to be connected to one of the points $P_0, \ldots, P_{n-1}$. We define the order σ of labels in C so that the point P_i is connected to the label $L_{\sigma(i)}$. The badness function of the cluster at position s_0 is then $\sum_i f_i(s_0 + \sigma(i) \cdot h)$.

In Lemma 5, we saw that the functions f_i are piecewise linear, and the endpoints of the segments correspond to either a horizontal-straight leader or a downward-straight leader.

Since the minimum of a sum of piecewise linear functions is attained at one of its extreme points, we conclude that there exists a minimal cluster labeling with a horizontal-straight leader or a downward-straight leader. □

3.1.2 Minimal Cluster Labeling using a Matching Algorithm

In this part, we use the matching algorithms defined in the preliminaries to compute a minimal cluster labeling. We are given a set of n points $P_0, \ldots, P_{n-1}$, a set of n labels $L_0, \ldots, L_{n-1}$ and we look for a minimal labeling supposing that all labels are placed into the same cluster.

We already know from Corollary 1 that there exists a minimal cluster labeling containing a horizontal-straight leader or a downward-straight leader. Given a pair (P_i, L_j), there exists a single cluster positions so that:

- The i bottommost point is P_i,
- The j bottommost label of the cluster is L_j,
- the point P_i is connected to L_j by a horizontal-straight leader.

The same holds for a downward-straight leader. Therefore, Corollary 1 provides $2 \cdot n^2$ cluster positions, and at least one of them is the position of a minimal cluster labeling.

Moreover, given a position of the cluster, the position of each of the n labels is directly implied. Therefore, Theorem 2 states that the Algorithm of Vaidya [Vai89] computes in

$O\big(n^2 \cdot (\log n)^3\big)$ a minimal cluster labeling. We construct the following algorithm.

Algorithm 3: Minimal Cluster WAB-Labeling for single-edge border with slope $m > 1$.

```
Data: {P_1, ..., P_{n-1}}
Result: Minimal Cluster Labeling: σ, L_0, ..., L_{n-1}
min_badness := null;
best_cluster := null;
best_σ := null;
foreach i ∈ {0 ... n − 1} do
    foreach j ∈ {0 ... n − 1} do
        Cluster C_hs := cluster position so that ℓ(P_i, L_j) would be horizontal-straight;
        (σ, badness) := OptimalMatching(C_hs);
        if badness < min_badness then
            min_badness := badness;
            best_σ := σ;
            best_cluster := C_hs;
        ;
        Cluster C_ds := cluster position so that ℓ(P_i, L_j) would be downward-straight;
        (σ, badness) := OptimalMatching(C_ds);
        if badness < min_badness then
            min_badness := badness;
            best_σ := σ;
            best_cluster := C_ds;
return best_σ, best_cluster;
```

Theorem 4. *Algorithm 3 computes in $O\big(n^4 \cdot (\log n)^3\big)$ time a minimal cluster labeling.*

Proof. The correctness and the running time follow directly from Corollary 1 and Theorem 2. Algorithm 3 computes $O(n^2)$ matchings that are computed in $O\big(n^2 \cdot (\log n)^3\big)$ time, and one of theses matchings induces a minimal cluster labeling. □

Moreover, we can reduce the number of matching computations. A minimal cluster labeling containing a downward-straight also contains also an upward leader or an horizontal-straight leader. Indeed, for a single edge border with a big slope, moving upward a label connected to a downward leader decreases its leader length, because the leader length function is convex and the minimal leader length is attained for a horizontal-straight leader. Thus, moving upward a cluster only containing downward leaders decrease the badness function.

For this reason, every cluster position computed for a downward-straight leader but only containing downward leaders cannot induce a minimal cluster labeling. Every cluster so that $s_0 < y_0 - n \cdot h$ is so that $s_j < y_i$ for every pair (P_i, L_j) of point and label. These cluster positions can thus not induce a minimal labeling.

Finally, depending on the slope of the edge and the position of the points, we may be able to compute only n^2 matchings instead of $2 \cdot n^2$, speeding up the algorithm by a factor 2.

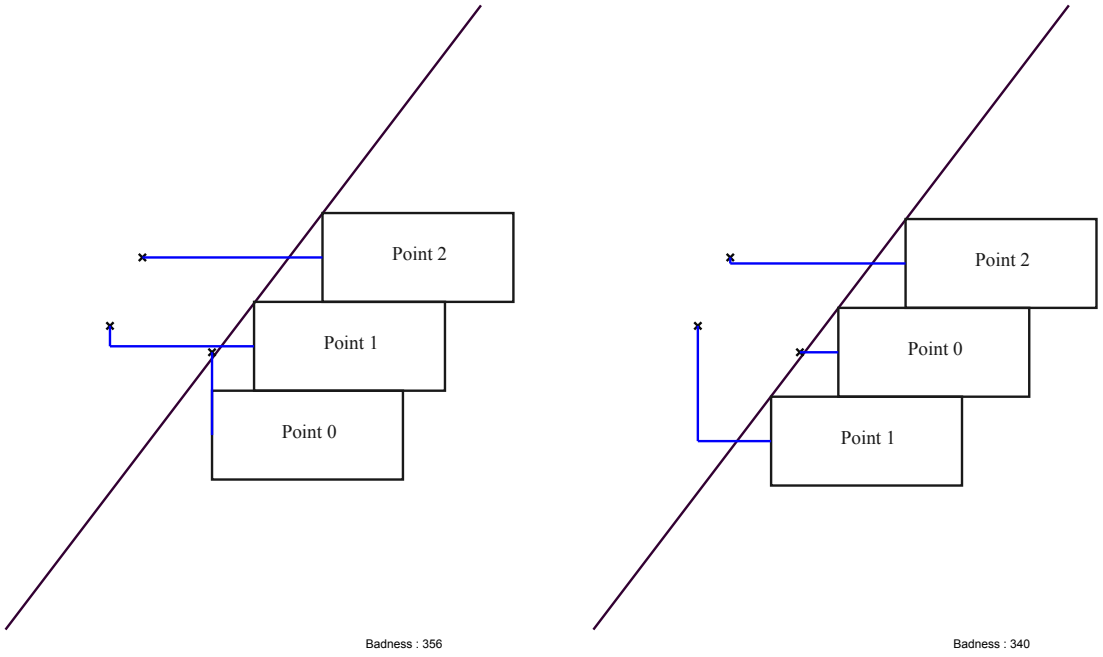
3.1.3 Minimal Cluster Labeling using a Sweep-line Algorithm

In this part, we still suppose that every labels has to be grouped into the same cluster, and we look for a minimal cluster labeling, that is an order of the labels and a position

of the whole cluster which minimize the total leader length. We consider that the slope is positive and big: $m > 1$.

In the previous part, we already saw an algorithm to compute a minimal cluster labeling in $O\big(n^4 \cdot (\log n)^3\big)$ time, which is a much more than the time complexity provided by Lemma 3 in the case of a vertical border. This lemma states that a minimal cluster labeling for n labels can be computed for a vertical boundary in $O(\log n)$ time by calculating the median of n elements. In this part, we will generalize this algorithm to speed-up the computation of a minimal cluster position.

We recall the two main results for a vertical border, i.e., when the slope $m > 1$ is equal to $+\infty$. First, there exists a minimal labeling using the Y-order. Moreover, for the Y-order, the cluster has a minimal total leader length when at most half of the leaders are upward and at most half of them are downward. Unfortunately, things do not go so well for a slanted border. Indeed, some labels can be shifted. Therefore, it is possible to reduce the total leader length by switching a shifted label with another label left of it so that no labels are shifted afterward. This way, we decrease the horizontal leader length. Figure 3.4 shows a counter-example where the minimal cluster labeling with the Y-order contains a shifted label, and switching it with another label reduces the total leader length.



(a) optimal labeling using Y-order (b) optimal labeling

Figure 3.4: Example where the Y order is suboptimal. Switching the labels L_0 and L_1 reduces the total leader length.

Therefore, the Y-order may be suboptimal, and we have to find not only the optimal position of a cluster, but also an order of the labels which gives a minimal labeling.

3.1.3.1 Position of a Cluster with Fixed Order

We suppose first that we know in which order the labels have to be positioned in the cluster. We are looking for a cluster position which minimizes the leader length.

We are given n points $P_0, \ldots, P_{n-1}$ sorted by increasing ordinates. The coordinates of a point P_i are denoted by (x_i, y_i). We consider a cluster of n labels $L_0, \ldots, L_{n-1}$, the label

L_i has the coordinates (t_i, s_i). These labels are grouped into a single cluster and sorted by increasing ordinates: $s_0 = s_1 - h = \ldots = s_{n-1} - (n-1) \cdot h$, where h is the height of a label. We denote by σ the bijection between the points and the labels, i.e., the point P_i is connected to the label $L_{\sigma(i)}$.

We know from Corollary 1 that there exists a minimal cluster labeling with a downward-straight leader or an horizontal straight leader. Since we know in which order the labels are positioned, each horizontal-straight leader and each downward-straight leader induce a position for every labels Therefore, Corollary 1 induces $2n$ cluster positions, and one of them corresponds to a minimal cluster labeling. Moreover, for each of those cases, we can compute in linear time the position of each label and the total leader length associated to this cluster labeling. Thus, we can compute a position of the cluster with minimal badness for the given order in $O(n^2)$ time.

If the minimal cluster labeling for the given order contains no shifted label, Algorithm 4 computes the optimal position faster:

Algorithm 4: Possibly Minimal Cluster WAB-Labeling for a given order of labels, for single-edge border with slope $m > 1$

Data: σ, $\{P_0, \ldots, P_{n-1}\}$
Result: A possibly minimal cluster labeling
$Y' = \emptyset$;
foreach $i \in \{0 \ldots n-1\}$ **do** add $(y_i - \sigma(i) \cdot h, i)$ in Y' ;
sort Y' by increasing $y_i - \sigma(i) \cdot h$ value.;
$\nu =$ unique integer value in $[\frac{n}{2}(1 - \frac{1}{m}), \frac{n}{2}(1 - \frac{1}{m}) + 1[$;
$(y_k - \sigma(k) \cdot h, k) = \nu^{th}$ element in Y';
return cluster position so that $\ell(P_k, L_{\sigma(k)})$ is horizontal-straight;

Note that when $m = \infty$, the border is vertical and $\nu \in [\frac{n}{2}, \frac{n}{2} + 1[$. Then k corresponds to the median of Y'. Here, when the border has a positive finite slope, the cluster tends to be pushed downward. If $m \approx 1$, the length of a leader does not change much when the label is moved below its optimal position, but increase a lot when it is pushed above its optimal position. Therefore, the optimal position of the cluster will contain only straight and downward leaders.

Lemma 6. *If the cluster position computed in Algorithm 4 contains no shifted label, then this position has a minimal badness among the cluster positions with the given order σ.*

Proof. This proof is composed of two steps. We first consider the virtual badness function where the horizontal length of a leader $\ell(P_i, L_{\sigma(i)})$ is given by $t_{\sigma(i)} - x_i$ instead of $|t_{\sigma(i)} - x_i|$. Therefore, there will be no shifted label, and if the left side of a label moves left of the point it is connected to, the horizontal leader length will be negative. We compute the minimal cluster labeling for this case.

Then, we prove that this cluster position is optimal for the real badness function if every horizontal leader length is positive, i.e., if this cluster position induces no shifted label using the real badness function.

We suppose now that the horizontal length of a leader i is $t_{\sigma(i)} - x_i$. The total leader length depending on s_0 is given by the following badness function:

$$F_1(s_0) = \sum_{i=0}^{n-1} \big(|s_{\sigma(i)} - y_i)| + (t_{\sigma(i)} - x_i)\big)$$

We replace $s_{\sigma(i)}$ and $t_{\sigma(i)}$ by their expression $s_{\sigma(i)} = s_0 + \sigma(i) \cdot h$ and $t_{\sigma(i)} = a + \frac{s_{\sigma(i)}}{m} = a + \frac{s_0 + \sigma(i) \cdot h}{m}$:

$$F_1(s_0) = \sum_{i=0}^{n-1} \left(|y_i - \sigma(i) \cdot h - s_0| + \frac{s_0}{m} + \frac{\sigma(i) \cdot h}{m} + a - x_i \right)$$

Since x_i, a, h and m are independent of s_0, minimizing F_1 is the same as minimizing:

$$F_2(s_0) = \sum_{i=0}^{n-1} \left(|(y_i - \sigma(i) \cdot h) - s_0| + \frac{s_0}{m} \right) = s_0 \cdot \frac{n}{m} + \sum_{i=0}^{n-1} \left(|(y_i - \sigma(i) \cdot h) - s_0| \right)$$

Let's look for local minima of the function F_2. We fix a position s_0 for the bottommost cluster and a value $\varepsilon > 0$. We define the gain function upward $G_u(s_0, \varepsilon) = F_2(s_0 + \varepsilon) - F_2(s_0)$ and the gain function downward $G_d(s_0, \varepsilon) = F_2(s_0 - \varepsilon) - F_2(s_0)$. Let's remember that the position s_0 for the bottommost label induces a locally optimal labeling if both gains are positive for any value of ε. From Lemma 5, we know that the total leader length function is convex, and therefore a local minimum is a global minimum.

Let $\Delta_i(s_0, \varepsilon) = |(y_i - \sigma(i) \cdot h) - (s_0 + \varepsilon)| - |(y_i - \sigma(i) \cdot h) - s_0|$. We have then:

$$G_u(s_0, \varepsilon) = \varepsilon \frac{n}{m} + \sum_{i=0}^{n-1} \Delta_i(s_0, \varepsilon)$$

$$G_d(s_0, \varepsilon) = -\varepsilon \frac{n}{m} + \sum_{i=0}^{n-1} \Delta_i(s_0, -\varepsilon)$$

Given $i \in \{0, 1, \ldots, n-1\}$ and $\varepsilon > 0$, $\Delta_i(s_0, \varepsilon) = \begin{cases} +\varepsilon & \text{if} \quad y_i - \sigma(i) \cdot h \leq s_0 \\ -\varepsilon & \text{if} \quad s_0 \leq y_i - \sigma(i) \cdot h - \varepsilon \end{cases}$

Similarly, $\Delta_i(s_0, -\varepsilon) = \begin{cases} -\varepsilon & \text{if} \quad y_i - \sigma(i) \cdot h + \varepsilon \leq s_0 \\ +\varepsilon & \text{if} \quad s_0 \leq y_i - \sigma(i) \cdot h \end{cases}$

Since we look for a local minima, we can choose ε small enough so that $\Delta_i(s_0, \varepsilon) = \pm\varepsilon$ for each value of i. Then, $\Delta_i(s_0, \varepsilon) = +\varepsilon$ if the leader incident to the label L_i is upward or straight, and $\Delta_i(s_0, \varepsilon) = -\varepsilon$ if the leader is downward. We define now:

- $n_s = |\{i \mid s_0 = y_i - \sigma(i) \cdot h\}|$ the number of straight leaders.
- $n_d = |\{i \mid s_0 < y_i - \sigma(i) \cdot h\}|$ the number of downward leaders.
- $n_u = |\{i \mid s_0 > y_i - \sigma(i) \cdot h\}|$ the number of upward leaders.

We have then $n_s + n_d + n_u = n$ and:

- $G_d(s_0, \varepsilon) = (-\frac{n}{m} + n_s - n_u + n_d)\varepsilon = (n - \frac{n}{m} - 2 \cdot n_u)\varepsilon$.
- $G_u(s_0, \varepsilon) = (\frac{n}{m} + n_s + n_u - n_d)\varepsilon = (\frac{n}{m} - n + 2 \cdot n_s + 2 \cdot n_u)\varepsilon$.

We have a local minimum when there exists a value $\delta > 0$ so that $G_d(s_0, \varepsilon) \geq 0$ and $G_u(s_0, \varepsilon) \geq 0$ for each positive value $\varepsilon < \delta$.

- $G_d(s_0, \varepsilon) \geq 0 \Rightarrow n_u \leq \frac{1}{2}(n - \frac{n}{m})$.
- $G_u(s_0, \varepsilon) \geq 0 \Rightarrow n_u \geq \frac{1}{2}(n - \frac{n}{m}) - n_s$.

From Corollary 1, we know that there exists an optimal position with a horizontal-straight leader or a downward-straight leader. We supposed here that $x_i < t_{\sigma(i)}$ for each i. This means that we cannot have any downward-straight leader. Thus, this minimal cluster labeling must contain a horizontal-straight leader.

Let's consider an optimal position of the cluster with $n_s \geq 1$ straight leaders and $n_u \in [\frac{n}{2}(1-\frac{1}{m}) - n_s,\ \frac{n}{2}(1-\frac{1}{m})]$ upward leaders. If $\frac{n}{2}(1-\frac{1}{m}) - n_s$ is an integer value, then $\frac{n}{2}(1-\frac{1}{m}) - n_s + 1$ is another integer value for n_u corresponding to an minimal cluster labeling. Therefore, we can exclude the value $\frac{n}{2}(1-\frac{1}{m}) - n_s$ from the interval of values for n_u. We suppose now:

$$\frac{n}{2}(1-\frac{1}{m}) - n_s < n_u \leq \frac{n}{2}(1-\frac{1}{m}) \tag{1}$$

The elements in Y' are sorted by increasing value of $y_i - \sigma(i)\cdot h$, then the n_u first elements of Y' correspond to upward leaders with $y_i < s_{\sigma(i)} = s_0 + \sigma(i)\cdot h$, and the n_d last elements of Y' correspond to downward leaders with $y_i > s_{\sigma(i)} = s_0 + \sigma(i)\cdot h$. The straight leaders corresponds to the elements from the index $n_u + 1$ to the index $n_u + n_s$.

For each $k_1 \in \{1\ldots,\ n_s\}$, we deduce from (1):

$$\frac{n}{2}(1-\frac{1}{m}) + k_1 - n_s < n_u + k_1 \leq \frac{n}{2}(1-\frac{1}{m}) + k_1 \tag{2}$$

Therefore, the n_s integer between $n_u + 1$ and $n_u + n_s$ are between $\frac{n}{2}(1-\frac{1}{m}) + 1 - n_s$ and $\frac{n}{2}(1-\frac{1}{m}) + 2\cdot n_s$. This interval contains exactly $2\cdot n_s + 1$ value. Since the n_s values $n_u + k_i$ are consecutive, we deduce that one of theses values is the n_s-th value of the interval. In others words:

$$\exists k \in \{1,\ldots,n_s\},\ \nu = n_u + k \in [\frac{n}{2}(1-\frac{1}{m}), \frac{n}{2}(1-\frac{1}{m}) + 1[\tag{3}$$

Finally, the position of the cluster so that the leader connected to the ν bottommost point is straight has a minimal leader length with the order of labels σ.

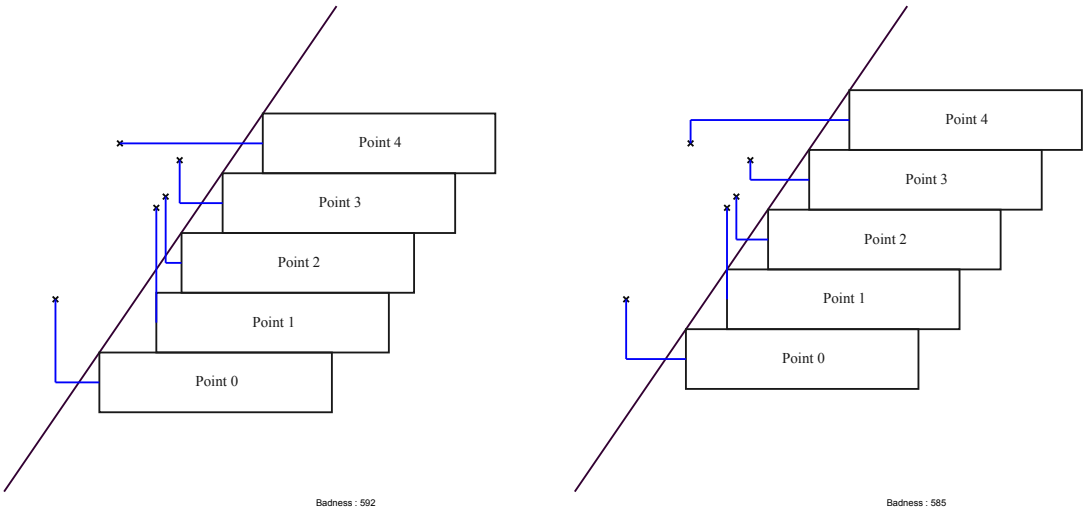
We just proved that Algorithm 4 would return the optimal cluster position if the badness function was $F_1(s_0) = \sum |y_i - s_{\sigma(i)}| + \big(x_i - t_{\sigma(i)}\big)$ instead of $F(s_0) = \sum |y_i - s_{\sigma(i)}| + |x_i - t_{\sigma(i)}|$. Consider now the minimal cluster labeling given by s_0^* for the badness F_1. There is a negative horizontal leader length for the badness function F_1 if and only if there is a shifted label for the badness function F. Therefore, we have to prove that this cluster position is optimal for the badness function bad when there is no negative horizontal leader length with the badness function F_1. Since each number is smaller or equal to its absolute value, we have for each s_0: $F_1(s_0) \leq \text{bad}(s_0)$. Because the position s_0^* induces no negative leader length, we have $\text{bad}(s_0^*) = F_1(s_0^*) = \min_{s_0} F_1(s_0) \leq \min_{s_0} F(s_0)$. We conclude that s_0^* is the minimal cluster labeling for the badness function bad and the order σ of labels, and thus that Algorithm 4 returns the minimal cluster labeling. □

Theorem 5. *Algorithm 4 computes a cluster position in $O(n \log n)$ time. The obtained labeling is optimal for the given order of labels if this cluster position contains no shifted label.*

Proof. The optimality of the cluster position is proven by Lemma 5. The algorithm computes the elements of Y' in linear time and sorts it in $O(n \log n)$ time. Then, the index of a point connected by a straight leader is found in constant time from Y', and finally, the position of each label is computed in linear time. Thus, the total running time in $O(n \log n)$. □

However, when the cluster position computed by Algorithm 4 contains a shifted label, this position may not be optimal. Figure 3.5 shows a counter-example.

In this case, we have to compare the total leader length of each of the $2n$ possible optimal positions, corresponding to the extreme points of the badness function $\sum_i f_i(s_0 + \sigma(i)\cdot h)$.



(a) Labeling computed by Algorithm 4 (b) Labeling computed by Algorithm 5

Figure 3.5: Example of minimal cluster labeling containing no horizontal-straight leader. The cluster labeling computed by Algorithm 4 is not minimal.

The total leader length of a cluster position is computed in linear time. Therefore, the optimal position of a cluster given the order of the labels can be computed in $O(n \log n)$ time, with the following algorithm.

Algorithm 5: Minimal Cluster Labeling for a Given Order of Labels, for single-edge border with slope $m > 1$

Data: σ, $\{P_0, \ldots, P_{n-1}\}$
Result: Minimal cluster labeling.
$Y' := \emptyset$;
foreach $i \in \{0 \ldots n-1\}$ **do**
  Add $(y_i - \sigma(i) \cdot h$,HS) in Y' ;
  Add $(z_i - \sigma(i) \cdot h$,DS) in Y', where z_i is the y-coordinate of the label L_i when its leader is downward straight.;
Sort Y' in decreasing order.;
new_s_0 := cluster position corresponding to the first element of Y';
new_ll := total leader length for position new_s_0;
n_s := number of horizontal-straight leaders for position new_s_0;
n_u := number of upward leaders for position new_s_0;
n' := number of downward-straight and downward-shifted leader for position new_s_0;
foreach *cluster position in* Y' **do**
  previous_s_0 := new_s_0;
  previous_ll:= new_ll;
  new_s_0 := y-coordinate of the cluster position;
  $n_u := n_u - n_s$;
  n_s := number of element corresponding to the next cluster position with second field with value HS;
  n' := n'+ number of element corresponding to the next cluster position with second field with value DS;
  new_ll := previous_ll + (previous_s_0 - new_s_0) $\cdot\left(n - 2 \cdot n_u - \frac{n-n'}{m}\right)$;
  update the best badness and the associated s_0 value.;
return the best labeling;

Theorem 6. *Algorithm 5 computes in $O(n \log n)$ time a minimal cluster labeling of the cluster for the given order of labels.*

Proof. The correctness of Algorithm 5 follows from Corollary 1. We now explain how to compute in constant time the badness of a cluster labeling at a position given the badness at the previous cluster position. From the previous cluster position to the new one, the cluster is moved upward by a distance of previous_s_0 - new_s_0. Moreover, no upward label become downward, and no downward-regular label become downward-shifted. Therefore, the vertical leader length is increased by (previous_s_0 - new_s_0)$\cdot(n_s + n_d - n_u)$. Since $n = n_s + n_d + n_u$, the vertical leader length is increased by (previous_s_0 - new_s_0)$\cdot(n - 2 \cdot n_u)$. Furthermore, every label connected to either an upward, or a horizontal-straight or a downward-regular leader reduces its horizontal leader length by (previous_s_0 - new_s_0)$/m$. In Algorithm 5, we set n' as the number of leaders that are downward-straight or downward-shifted in the previous cluster position and are downward-shifted in the new cluster position. Since the horizontal length of these n' labels do not change, the horizontal leader length is then reduced by (previous_s_0 - new_s_0)$\cdot(n - n')/m$. We conclude:

$$\text{new\_ll} := \text{previous\_ll} + (\text{previous\_}s_0 - \text{new\_}s_0) \cdot \big(n - 2 \cdot n_u - \frac{n - n'}{m}\big)$$

In order to update in constant time the values of n_u, n_s and n', we associate with each element of Y' an index telling if the element corresponds to a horizontal-straight leader event HS or to a downward-straight leader event DS. Since each leader can be only once downward-straight and only once upward-straight, the update of n_u, n_s and n' is done in amortized constant time. Therefore, the sweep-line is computed in linear time, and the time complexity of Algorithm 5 is bounded by the $O(n \log n)$ time required to sort the list Y'. □

3.1.3.2 Finding an Optimal Order

In the previous part, we saw algorithms to compute the optimal position of a cluster of n labels given the order of the labels σ. Now, we look for an order of the labels which provides a minimal labeling. We already saw an example where the Y-order does not provide a minimal labeling. The following lemma gives a sufficient condition so that the cluster-labeling computed by Algorithm 4 is minimal.

Lemma 7. *If the cluster labeling computed by Algorithm 4 with the Y-order contains no shifted labels, then this cluster labeling has a minimal badness among every cluster labeling with any order of labels.*

Proof. Let $\mathcal{L}^*$ be the cluster labeling produced by Algorithm 4. Consider a labeling $\mathcal{L}_2$ of the same cluster with an order σ_2 for the labels and a cluster position α_2. We have to prove that this labeling has a badness greater or equal to the badness of $\mathcal{L}^*$.

We denote here by *projection* of a WAB-labeling $\mathcal{L}$ the labeling proj($\mathcal{L}$) computed by moving every shifted labels back to the border. The cluster labeling proj($\mathcal{L}$) is not a WAB-labeling if $\mathcal{L}$ contains shifted labels. We denote then by F_1 the badness function:

$$F_1(\mathcal{L}) = \sum_{i=0}^{n-1} \big(|s_{\sigma_{\mathcal{L}}(i)} - y_i)| + (t'_{\sigma_{\mathcal{L}}(i)} - x_i)\big),$$

where $t_{\sigma'_{\mathcal{L}}(i)}$ corresponds to the x-coordinate of the label $L_{\sigma'_{\mathcal{L}}(i)}$ in the labeling proj($\mathcal{L}$). We saw in the proof of Lemma 6 that Algorithm 4 computes a cluster labeling minimizing

the badness $F_1(\text{proj}(\mathcal{L}))$ of the projection of the cluster WAB-labelings $\mathcal{L}$. The badness to minimize is the total leader length $\text{bad}(\mathcal{L})$:

$$\text{bad}(\mathcal{L}) = \sum_{i=0}^{n-1} \left(|s_{\sigma_{\mathcal{L}}(i)} - y_i)| + |t_{\sigma_{\mathcal{L}}(i)} - x_i| \right)$$

Since $|t_{\sigma_{\mathcal{L}}(i)} - x_i| \geq (t_{\sigma_{\mathcal{L}}(i)} - x_i)$, we deduce:

$$\forall \mathcal{L},\ \text{bad}(\mathcal{L}) \geq F_1(\text{proj}(\mathcal{L}))$$

We have in particular:

$$\text{bad}(\mathcal{L}_2) \geq F_1(\text{proj}(\mathcal{L}_2)) \tag{1}$$

Since $\mathcal{L}^*$ contains no shifted label, we deduce:

$$\text{bad}(\mathcal{L}^*) = F_1(\text{proj}(\mathcal{L}^*)) \tag{2}$$

We create now from $\mathcal{L}_2$ a cluster labeling $\mathcal{L}_3$ by switching iteratively the labels to obtain the Y-order. The total horizontal leader length are the same for the projection of these two labeling because $\sum_{0 \leq i < n} t_i$ is then constant. Given a position of the cluster, we saw in Section 2.4 that the Y-order minimizes the vertical leader length. Therefore:

$$F_1(\text{proj}(\mathcal{L}_2)) \geq F_1(\text{proj}(\mathcal{L}_3)) \tag{3}$$

Since Algorithm 4 computes a cluster labeling minimizing the badness function F_1 of the projection of the cluster labelings using the Y-order, we have:

$$F_1(\text{proj}(\mathcal{L}_3)) \geq F_1(\text{proj}(\mathcal{L}^*)) \tag{4}$$

We deduce from (1), (2), (3) and (4):

$$\text{bad}(\mathcal{L}_2) \geq F_1(\text{proj}\,\mathcal{L}_2)) \geq F_1(\text{proj}(\mathcal{L}_3)) \geq F_1(\text{proj}(\mathcal{L}^*)) = \text{bad}(\mathcal{L}^*)$$

We conclude that the badness of the initial labeling $\mathcal{L}_2$ is greater or equal to the badness of the labeling $\mathcal{L}^*$ computed by Algorithm 4 using the Y-order. □

In addition, a similar result holds for a labeling created by Algorithm 5:

Lemma 8. *If the labeling computed by Algorithm 5 with the Y-order contains no shifted labels and no downward-straight leader, then this labeling has a minimal badness among every labeling with any order of labels.*

Proof. Let $\mathcal{L}^*$ be the cluster labeling returned by Algorithm 4 and $\mathcal{L}'$ be the cluster labeling returned by Algorithm 5, both using the Y-order. We suppose that $\mathcal{L}'$ contains no vertical leader, i.e., no downward-straight and no downward-shifted leader. We prove then that the position of the labelings $\mathcal{L}'$ and $\mathcal{L}^*$ are the same.

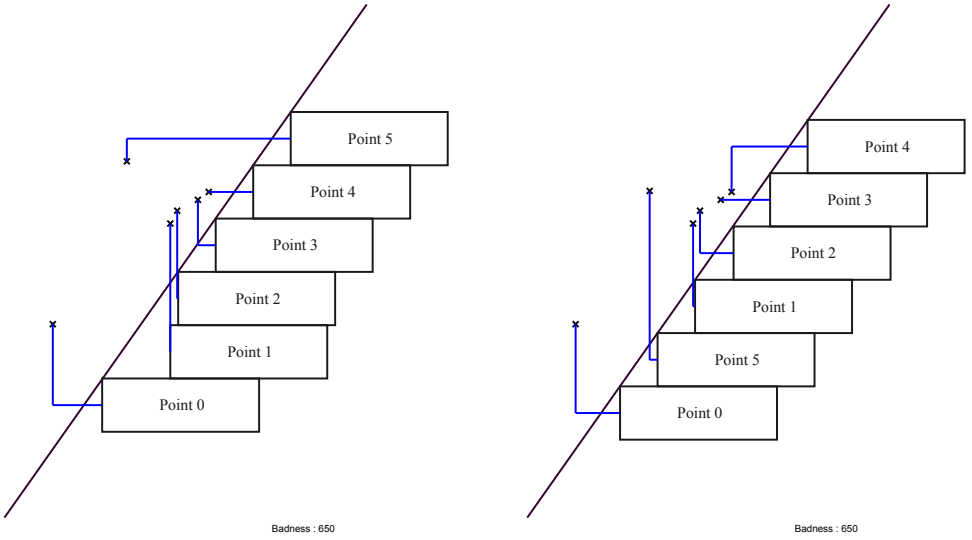
We use here the same notation as in the proof of Lemma 7. Since $\mathcal{L}'$ contains no shifted label, we have:

$$\text{bad}(\mathcal{L}') = F_1(\text{proj}(\mathcal{L}'))$$

Let s' be the position of the cluster labeling $\mathcal{L}'$. Since $\mathcal{L}'$ contains no downward-straight leader, there exists an interval of positions $]s' - \varepsilon,\ s' + \varepsilon[$ where every WAB-labeling contains no shifted label. In this interval, the badnesses $\text{bad}(\mathcal{L})$ and $F_1(\text{proj}(\mathcal{L}))$ are equal. Therefore, s' is a local minimum of for the badness function $F_1(\text{proj}(\mathcal{L}))$.

However, we can prove with the same reasoning as in Lemma 5 that the badness function $F_1(\text{proj}(\mathcal{L}))$ is convex. Since $\mathcal{L}^*$ is a minimum of this function, we have $\mathcal{L}^* = \mathcal{L}'$. We deduce with Lemma 7 that this cluster labeling is minimal. □

Suppose now that the labeling given by Algorithm 4 contains shifted labels. Figure 3.6 shows two similar maps. Both labelings are minimal. In the first figure, the Y-order provides a minimal labeling. On the second figure, the points have a similar position but the Y-order does not provide any minimal labeling. Indeed, on the second figure, the y-coordinates of the two points labeled 'Point 1' and 'Point 5' are close enough to each other so that connecting switching the two corresponding labels reduces more the horizontal leader length than increases the vertical leader length.



(a) First set of points: the Y-order is optimal

(b) Second set of points: the Y-order is not optimal

Figure 3.6: The two labelings above have exactly the same badness. From the left figure to the right one, the topmost point has been moved downward. As a result, positioning the label connected to the topmost point to the shifted label put this label back on the border without increasing the vertical leader length too much.

It is not easy to compute a minimal cluster labeling starting from the Y-order and by changing the order of labels to reduce the leader length.

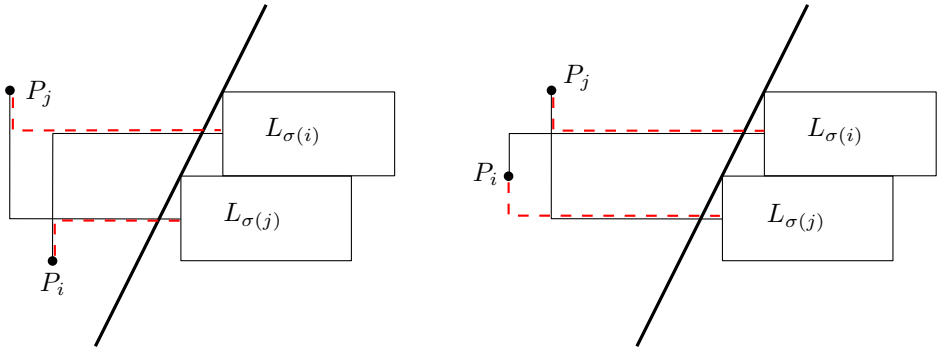
3.1.4 Removing the crossings

Now we will adapt the crossing-removal algorithm for a border composed of a single edge with positive slope. We suppose here that we have a minimal labeling which may contain leader crossings. We want to change some connections between labels and points to get an optimal labeling, without crossing. For a vertical border, we saw a $O(n^2)$ time algorithm [BHKN09] to remove crossings, where n is the number of labels.

The following lemma is a generalization of Lemma 4.

Lemma 9. *In a minimal labeling, there are no crossings between an upward leaders and a downward leader. Moreover, no straight leader can be involved in a crossing with an upward leader and a crossing with a downward leader at the same time.*

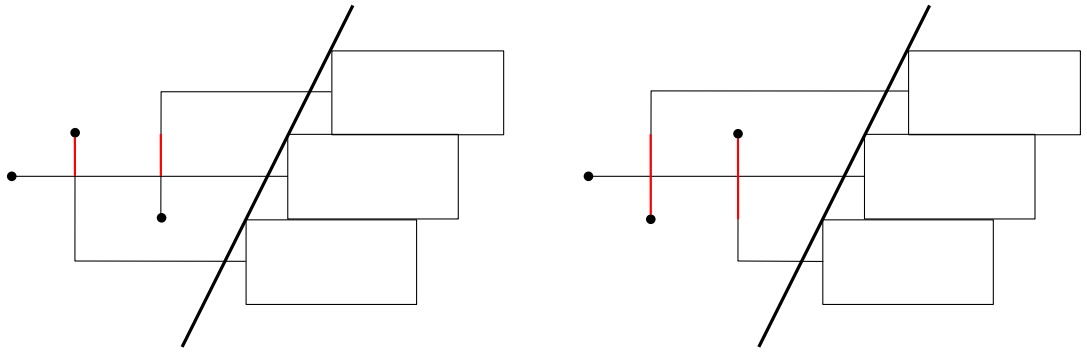
Proof. Suppose first that an upward leader $\ell(P_i, L_{\sigma(i)})$ crosses a downward leader $\ell(P_j, L_{\sigma(j)})$. The crossing is left of both labels, therefore switching the two labels will not create a new shifted label. Moreover, switching the labels reduces the vertical leader length, as presented in Figure 3.7. Therefore, the labeling is not minimal.



(a) First possibility (b) Second possibility

Figure 3.7: Two possibilities for a crossing between an upward an a downward label, depending on the segment of each leader involved in the crossing. In both cases, the labeling do not have a minimal badness. The dashed segments represent the leader after switching the labels.

The same reasoning can be done if a straight leader crosses both an upward leader and a downward leader. Consider then the leftmost crossing. Switching the two labels concerned does not change the badness function, but creates a crossing between an upward and a downward leader, as presented in Figure 3.8 □



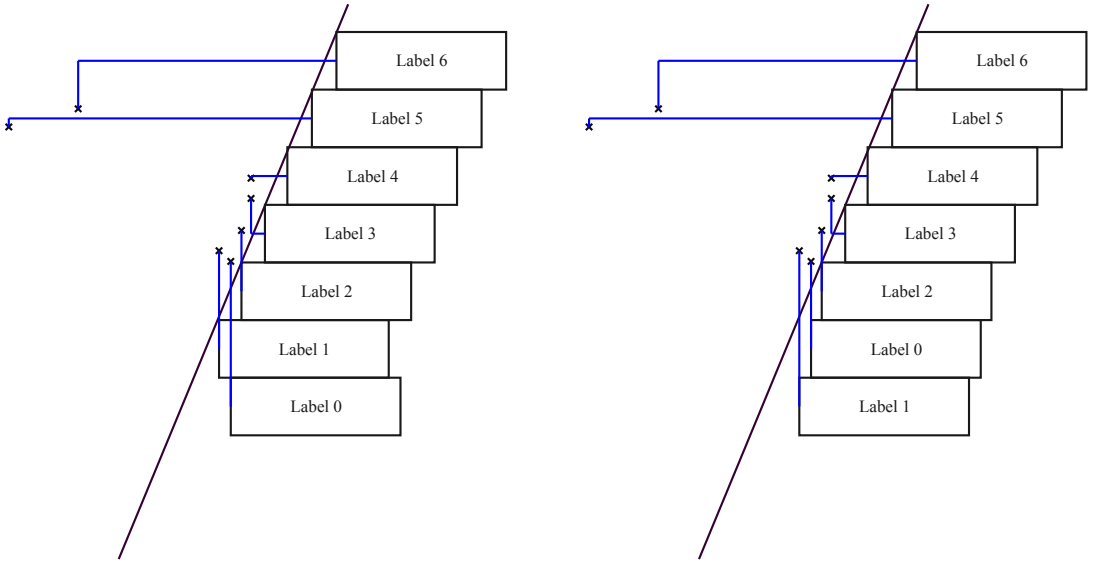
(a) First possibility (b) Second possibility

Figure 3.8: Examples of labeling where a straight leader crosses both an upward and a downward crossing. The highlighted strokes represent the reduction of the leader length.

Therefore, Algorithm 2 removes the leader crossings. However, in addition to leader crossings, we may here have intersections between a leader and a label. See Figure 3.9 for an example.

Lemma 10. *In a minimal labeling, a crossing between a label and a leader can only occur if the two concerned leaders are vertical, i.e., each of the two concerned labels has the same x-coordinate as the point it is connected to. Moreover, in a minimal labeling, a downward-straight leader cannot cross a downward-regular leader and be connected to a crossed label at the same time.*

Proof. Suppose that a leader $\ell(P_i, L_{\sigma(i)})$ crosses a label $L_{\sigma(j)}$, ad presented in Figure 3.10 Then, we have $\sigma(i) < \sigma(j)$, i.e., the label $L_{\sigma(j)}$ is above $L_{\sigma(i)}$. Because of the crossing, the



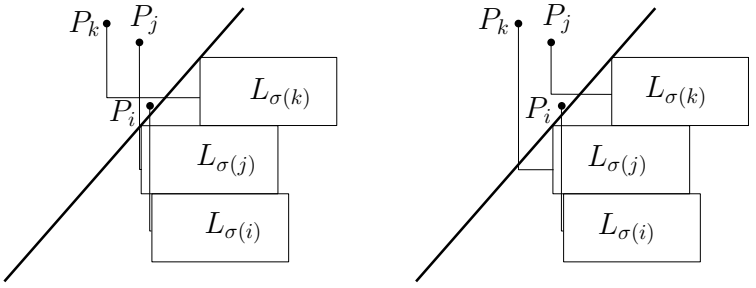
(a) Crossing (b) Resolution

Figure 3.9: Leader-label Crossing. Both concerned leaders are vertical.

label $L_{\sigma(i)}$ must be right of the label $L_{\sigma(j)}$. However, if the label $L_{\sigma(i)}$ was on the border, it would be left of the label $L_{\sigma(j)}$. Therefore, the label $L_{\sigma(i)}$ is shifted, and the leader $\ell(P_i, L_{\sigma(i)})$ is vertical.

Suppose now that the leader $\ell(P_j, L_{\sigma(j)})$ is not vertical, which means $x_j < t_{\sigma(j)}$. Then, switching the two labels reduces the horizontal leader length by either $t_{\sigma(i)} - x_i$, or the distance between the current position of $L_{\sigma(i)}$ and the position of a label with y-coordinate $s_{\sigma(i)}$ when on the border, depending on the x-coordinate x_j. The vertical leader length would either remain unchanged, when the leader $\ell(P_j, L_{\sigma(j)})$ is also downward, or would be reduced if this leader is upward. Then, the labeling would not be minimal. Thus the leader $\ell(P_j, L_{\sigma(j)})$ is vertical.

Consider finally a downward-shifted leader $\ell(P_i, L_{\sigma(i)})$ crossing a label $L_{\sigma(j)}$ so that the leader $\ell(P_j, L_{\sigma(j)})$ is downward-straight and crosses a regular leader $\ell(P_k, L_{\sigma(k)})$.



(a) Initial Labeling (b) After swithing $L_{\sigma(j)}$ and $L_{\sigma(k)}$

Figure 3.10: Leader-label Crossing. Both concerned leaders are vertical.

Therefore, the point P_k is left of the point P_j and switching the labels $L_{\sigma(j)}$ and $L_{\sigma(k)}$ does not change the total leader length. After the switch, the point P_k is connected to the label $L_{\sigma(j)}$, itself crossed by the leader $\ell(P_i, L_{\sigma(i)})$. However, the leader $\ell(P_k, L_{\sigma(j)})$ is regular. We saw at the beginning of this proof that, if the leader $\ell(P_k, L_{\sigma(j)})$ is not vertical, then the labeling is not minimal. □

Algorithm 6 solves leader-label crossings. It runs the same way as Algorithm 2.

Algorithm 6: Leader-Label Crossings Removal for single-edge border with slope $m > 1$.

Data: σ and $L_j : (t_j, s_j)$ for a minimal labeling, $P_i : (x_i, y_i)$
Result: Remove the leader-label crossings
foreach j *from* $n-1$ *to* 0 **do**
  $i = \sigma^{-1}(j)$.;
  if L_j *is crossed by a leader* **then**
    $L_k :=$ label connected by a downward-straight leader having the rightmost leader-label crossing with L_j;
    Switch the position of L_j and L_k.;

Theorem 7. *Given a minimal labeling, Algorithm 6 removes in $O(n^2)$ time the leader-label crossings and Algorithm 2 removes in $O(n^2)$ time the leader crossings.*

Proof. We know that Algorithm 2 removes in $O(n^2)$ time the leader-crossings inside the map. Algorithm 6 removes leader-label crossings in $O(n^2)$ time by switching labels. Indeed, Algorithm has been created so that after the j-th step of the iteration, the labels $L_j, \ldots, L_{n-1}$ do not cross any leader. Since the j-th step does not involve the labels $L_{j+1}, \ldots, L_{n-1}$, the crossings involving label L_j are to the left of the labels $L_{j+1}, \ldots, L_{n-1}$, and therefore switching two labels at the j-th step does not create any crossing with one of the labels $L_{j+1}, \ldots, L_{n-1}$. Thus, at the end of the algorithm, there is no leader-label crossing.

Since the switched labels are always incident to vertical leaders, this algorithm only changes segments of leaders inside the map. Therefore, Algorithm 6 creates no new leader-crossings.

We conclude that these two algorithms compute in quadratic time an optimal crossing-free labeling from a minimal labeling.

Moreover, Lemma 10 states that in a minimal labeling no leader can be involved in both a leader crossing and a leader-label crossing. Therefore, the two algorithms can be run in an arbitrary order. □

3.1.5 Computing an Optimal Labeling

In the previous parts, we studied how to compute a minimal cluster labeling, and how to remove crossings in a minimal labeling. In this part, we study now the Upward-downward algorithm presented in Section 2.4 in order to compute a minimal labeling from algorithms producing minimal cluster labelings.

First of all, we regroup Algorithm 4 and Algorithm 3 into a single algorithm:

Algorithm 7: Minimal Cluster WAB-Labeling for single-edge border with slope $m > 1$

Data: Points $P_0, \ldots, P_{n-1}$ corresponding to the labels of a cluster.
Result: Minimal cluster labeling.
Use Algorithm 4 with $\sigma :=$ Y-order to compute a position of the labels $L_0, \ldots, L_{n-1}$;
if $\exists j \mid L_j$ *shifted* **then** return the labeling computed by Algorithm 3;
else return (Y-order, $L_0, \ldots, L_{n-1}$);

Now, we prove with the following lemma that the Upward downward algorithm computes a minimal cluster labeling.

Lemma 11. *If the length functions of the leaders are convex, then the Upward-downward algorithm (Algorithm 1). To use this algorithm, we must be able to compute a minimal cluster labeling.*

Proof. The upward-downward algorithm first positions the label connected to P_0 at its optimal position. Then, for each point P_i considered with respect to their increasing y-coordinates:

1. Add the label connected to P_i at its optimal position. This label can be considered as a cluster above the others.
2. While this cluster intersects the cluster below it, merge the two clusters.

The upward-downward algorithm can be interpreted as follows. We position each label at its optimal position, and then consider overlapping labels. When two clusters intersect, they have to be regrouped into the same cluster, and two regrouped cluster never have to be separated afterward.

We prove that this assertion is true when the leader length functions are convex: Consider two clusters C_1 and C_2 containing respectively n_1 and n_2 labels, and suppose that we are at a point of the algorithm where these two clusters intersect. The clusters are constructed so that each point connected to a label in the bottom cluster C_1 is below the points connected to a label in the top cluster C_2. Let $f_{C_1}(s)$ and $f_{C_2}(s)$ be the length functions of the leaders connected to C_1 and C_2 given the y-coordinate of the bottommost label of the cluster. Let s_{C_1} and s_{C_2} be the computed minimal positions of the clusters. Because of the position of the clusters, we have $s_{C_1} < s_{C_2}$. Let C be the merger of the two clusters C_1 and C_2. The total leader length function associated to the cluster C is $f_C(s) = f_{C_1}(s) + f_{C_2}(s + n_1 \cdot h)$.

We know from Lemma 5 that the functions $f_i(s)$ are convex. Therefore the functions $f_{C_1}(s)$ and $f_{C_2}(s)$ are convex as sums of convex functions. The optimal position of the cluster C is $s_C \in [s_{C_2} - n_1 \cdot h, s_{C_1}]$. Indeed, consider a value $s > s_{C_1}$:

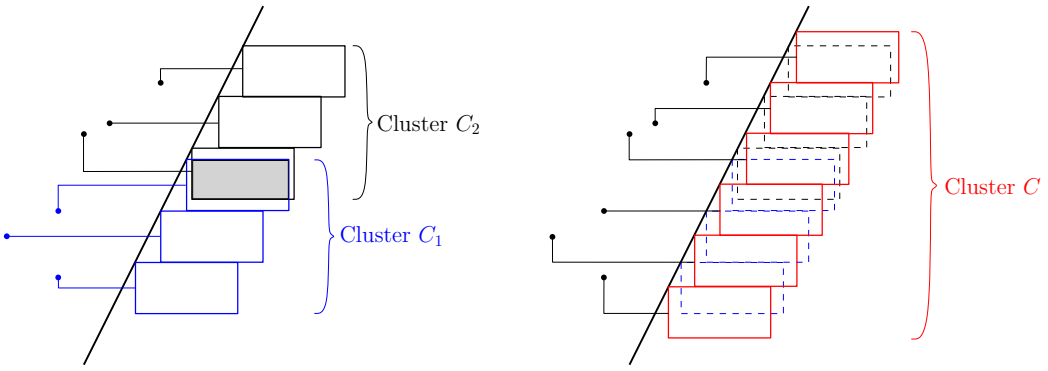
$$f_C(s) = f_{C_1}(s) + f_{C_2}(s + n_1 \cdot h)$$

Since s_{C_1} is the argument of the minimum of f_{C_1}, we have $f_{C_1}(s) > f_{C_1}(s_{C_1})$. Moreover, $s + n_1 \cdot h > s_{C_1} + n_1 \cdot h > s_{C_2}$ because the clusters intersect. Thanks to the convexity of the function f_{C_2}, we deduce: $f_{C_2}(s + n_1 \cdot h) > f_{C_2}(s_{C_1} + n_1 \cdot h) > f_{C_2}(s_{C_2})$. Finally, we have:

$$f_C(s) > f_{C_1}(s_{C_1}) + f_{C_2}(s_{C_1} + n_1 \cdot h) = f_C(s_{C_1})$$

This inclusion $s_C \in [s_{C_2} - n_1 \cdot h, s_{C_1}]$ means that every y-coordinate occupied by a label in the computed minimal positions of C_1 or C_2 is also occupied in the computed optimal position of C. Therefore, the labels of C_2 are pushed upward and the labels of C_1 are moved downward from their current position. In particular, the optimal position of each of the two clusters C_1 and C_2 has been moved away from its optimal position. Because of the convexity of the functions f_{C_1} and f_{C_2}, separating these clusters would increase the badness of the labeling. The same reasoning can be done for any partition $\{C_a, C_b\}$ of the cluster C, because their optimal positions would overlap.

Thus, moving one or several labels outside of the cluster at its optimal position would place them farther away from their optimal position corresponding to horizontal-straight leaders. Because of the convexity of the leader length functions, moving the labels closer to the cluster would then reduce the leader length. This shows that labels which were once merged into a cluster never have to be separated again.



(a) Overlapping clusters C_1 and C_2

(b) Cluster C entirely overlaying the clusters C_1 and C_2

Figure 3.11: When the leader length functions are convex, the optimal position of the new cluster C entirely overlay the position of the initial clusters C_1 and C_2.

Finally, we recall that two intersecting clusters are merged into a single cluster to prevent the label from intersecting. We proved here that not merging these clusters cannot reduce the total leader length. We conclude that the Upward downward algorithm computes a minimal cluster labeling. □

We conclude this part by providing a theorem combining the upward-downward algorithm (Algorithm 1), Algorithms 2 and 10 to remove crossings, and Algorithm 7 to compute a minimal cluster labeling, in order to compute an optimal cluster labeling:

Algorithm 8: Optimal WAB-Labeling for single-edge border with slope $m > 1$

Data: Points $P_0, \ldots, P_{n-1}$
Result: optimal labeling
Compute a minimal cluster labeling with Algorithm 1, which uses Algorithm 7 to compute minimal cluster labelings;
Call Algorithm 2 to remove leader crossings;
Call Algorithm 6 to remove leader-label crossings;
return the labeling;

Theorem 8. *Algorithm 8 computes an optimal labeling with a minimum time complexity of* $O\big(n^2\big)$ *and with a worst case complexity of* $O\big(n^5 \cdot (\log n)^3\big)$.

Proof. The correctness of Algorithm 8 follows from 11, the correctness of the algorithms computing minimal cluster labeling and from Theorem 7. In the best case, the cluster labelings computed by Algorithm 4 are minimal, and therefore we can compute in $O(n \log n)$ time a minimal cluster labeling. The Upward downward algorithm computes a minimal labeling from $O(n)$ cluster labelings. Since we only call Algorithm 4 with the Y-order, we can use an array structure for the list $Y' = \{y_i - i \cdot h\}$ to add in $O(n)$ time an element and compute then in constant time a minimal cluster labeling with Algorithm 4. In the best case, a minimal labeling is then created in $O(n^2)$ time.

In the worst case, we call $O(n)$ times Algorithm 3 with a $O\big(n^4 \cdot (\log n)^3\big)$ time complexity. The worst case complexity is then $O\big(n^5 \cdot (\log n)^3\big)$. □

3.1.6 Linear Programming

In this part, we present the minimal labeling problem as an Integer Linear Program (ILP). The objective is to compute with ab ILP solver minimal labelings in order to verify that the implementation of our algorithm creates a minimal labeling.

Given a set of n points, the equation $t(s) = a + s/m$ of the labels on the border and the height h of the labels, we create an ILP instance.

We recall the general form of an ILP:

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{subject to } Ax \leq b \\ &\qquad\qquad x \geq 0, \end{aligned}$$

Variables. The variables of the program are the position of the labels and the order of the labels. The position of the labels $L_0, \ldots, L_{n-1}$ are given by their y-coordinates $s_0, \ldots, s_{n-1}$:

$$\text{Variables: } S = \big(s_0, \ldots, s_{n-1}\big) \in \mathbb{R}^n \tag{3.1}$$

We consider here that the labels are sorted by increasing y-coordinates, and define the order of labels $\sigma(\cdot, \cdot)$ as the adjacency matrix of the points and the labels. Thus, given indexes i and j, $\sigma(i, j)$ is equal to 1 if the points P_i is connected to the label L_j, and 0 otherwise:

$$\text{Variables: } \sigma(i,j) \in \{0,1\}^{n^2} \text{ for } 0 \leq i, j \leq n-1 \tag{3.2}$$

In addition, we define further variables corresponding to the function to minimize. The constraints of the problem will define the relation between the variables above and these new variables. The function to minimize here is the sum of the leader lengths. We define two vectors LL_x and LL_y of n horizontal leader lengths and n vertical leader lengths. We denote by $LL_x[j]$ and $LL_y[j]$ the horizontal and vertical lengths of the leader incident to the label j.

$$Variables: \; LL_x = \big(LL_x[0], \ldots, LL_x[n-1]\big) \in \mathbb{R}^n \tag{3.3}$$

$$Variables: \; LL_y = \big(LL_y[0], \ldots, LL_y[n-1]\big) \in \mathbb{R}^n \tag{3.4}$$

Finally, we obtain the following vector of variables:

$$x = \Big(S, \sigma, LL_x, LL_y\Big)$$

Optimization Function. The badness function *bad* is easy to describe thanks to the variables LL_x and LL_y:

$$\text{minimize: } bad = \sum_{i=0}^{n-1} \Big(LL_x[i] + LL_y[i]\Big) \tag{3.5}$$

Constraints. The first constraints concerns the order of labels σ. Each label must be connected to a single point and each point must be connected to a single label:

$$\text{Constraints: } \forall j, \; \sum_i \sigma(i,j) = 1 \tag{3.6}$$

$$\text{Constraints: } \forall i, \; \sum_j \sigma(i,j) = 1 \tag{3.7}$$

Furthermore, the labels have to be right of the border described by the equation $t(s) = a + s/m$ of the labels $L : (t, s)$ on the border. This constraint it described by a lower bound of the length of the incident leader:

$$\text{Constraints: } \forall j,\ LL_x[j] \geq a + s_j/m - \sum_{i=0}^{n-1} \sigma(i,j) \cdot x_i \tag{3.8}$$

Moreover, the labels must be shifted to the right if their position on the border is left of the position of the point they are connected to. This constraint is introduced by defining the horizontal leader length $LL_x[j]$ of a leader $\ell(P_i, L_j)$ as the difference $t_j - x_i$ in x-coordinates of the label to the point. Then, this constraint is described as follows.

$$\text{Constraints: } \forall j,\ LL_x[j] \geq 0 \tag{3.9}$$

Therefore, the optimization function minimizes the horizontal leader length, and the horizontal length $LL_x[i]$ of a leader will be set equal to one of these two lower bounds, corresponding either to a label on the border or a shifted label.

The vertical length $|\ell(P_i, L_j)|$ of a leader is defined by $|s_j - y_i|$. Since the optimization function minimizes these lengths, we must prevent the variables $LL_y[j]$ to be greater than the actual length of their corresponding leader $length\big(\ell(P_{\sigma^{-1}(j)}, L_j)\big)$:

$$\begin{array}{l} \forall j,\ LL_y[j] \geq |\sum_{i=0}^{n-1}(\sigma(i,j) \cdot y_i) - s_j| \\ \forall j,\ -LL_y[j] \leq \sum_{i=0}^{n-1}(\sigma(i,j) \cdot y_i) - s_j \leq LL_y[j] \end{array}$$

Finally the constraints for the vertical leader length are the following:

$$\text{Constraints: } \forall j,\ LL_y[j] \geq \sum_{i=0}^{n-1} \Big(\sigma(i,j) \cdot y_i\Big) - s_j \tag{3.10}$$

$$\text{Constraints: } \forall j,\ LL_y[j] \geq s_j - \sum_{i=0}^{n-1} \Big(\sigma(i,j) \cdot y_i\Big) \tag{3.11}$$

The last Constraints for this problem is the non-overlapping of the labels. Each label $L_j : (t_j, s_j)$ occupies the vertical place from $y = s_j - h/2$ to $y = s_j + h/2$, and no other label can occupies a part of this place. Moreover, the labels $L_1, \ldots, L_{n-1}$ have been defined so that each label L_j is above the label L_{j-1}, for $j \neq 0$. Thus, the constraint of non-overlapping of labels is described by:

$$\text{Constraints: } \forall j \neq 0,\ s_j \geq s_{j-1} + h \tag{3.12}$$

When two consecutive labels L_j and L_{j+1} are grouped into the same cluster, we have then $s_j = s_{j-1} + h$.

3.1.7 Conclusion

In Section 3.1 , we studied the po-right-labeling problem when the border is composed of a single edge with positive inclination. For an edge with small slope $m \leq 1$, we can either replace the edge with a new edge of slope 1, or consider the edge as a part of the bottom side of the border. Moreover, in this case $m \leq 1$ the leader length functions contains two local minima corresponding to a downward-straight leader and to a horizontal-straight leader. Therefore, this problem can be considered as a form of two-sided labeling, where we have to decide for each label, if it will be placed to the right of the point or below the

point. Either way, this case has not been precisely treated. It might be an interesting issue to find out how to compute an optimal labeling in this case, and eventually considering a labeling with the bottommost side of the map.

Then, we saw a general algorithm using the algorithm of Vaidya [Vai89] for minimum weighted matching in bipartite complete graphs to compute a minimal cluster labeling in $O(n^4 \cdot (\log n)^3)$ time.

The generalization of the algorithm of the vertical border to compute a minimal cluster labeling is much faster, namely has a $O(n \log n)$ time complexity. However, it does not always provide a minimal labeling and no efficient method has been found to correct the sub-optimality due to shifted labels. However, the cluster labeling created is minimal if it contains no shifted label.

Furthermore, the algorithm for crossing removal used for a vertical border has been adapted by taking into account the new leader-label crossings. The time complexity stays in $O(n^2)$.

Then, we showed that the Upward-downward algorithm used with vertical border computes a clustering inducing a minimal labeling for the border consisting of a single edge with big slope $m > 1$.

Finally, we described an ILP Program to compute a minimal labeling. Thus, we can compare the results of the implementation of our algorithms with the results of the minimal labeling computed by an ILP solver to verify the correctness of our implementation.

3.2 Convex Boundary with Big Slopes

In this section, we present the generalization of the results and algorithms of the previous section for a convex border with big slopes. This section has exactly the same structure as Section 3.1. We first present the differences between a single edge boundary labeling and a labeling computer for a border with several edges, and generalize the integer linear program given in Section 3.1 into a integer quadratic program. Then, we give general results about the leader length functions and describe algorithms to compute a minimal cluster labeling. After that, we will see that the Upward-downward algorithm might not be optimal for a convex boundary consisting of several edges, and see how we could adapt it. Before concluding, we present the final algorithm computing an optimal labeling.

We consider here a convex border E composed of k edges $E_1, \ldots, E_k$. Figure 3.12 provides an example of an optimal labeling.

We saw in the previous section that, when the border is composed of an edge with positive slope, the top-left corner of every label is in contact with the border except shifted labels. Similarly, for a edge with negative slope, the bottom-left corner of the label must be on the border. When a convex border has edges with positive slopes and negative slopes, we must find a way to get from the top-left corner to the bottom-left corner of a label being on the border.

The boundary consists now of k edges $E_1, \ldots, E_k$ connected by the nodes $N_0, \ldots, N_k$. Let N_{right} be the rightmost node of the border. We denote by n_j the y-coordinate of a label so that the node N_j coincides with one of the corners of this labels. For $N_j = N_{right}$, we define a further y-coordinate n'_{right}, because in this case, both the top-left and the bottom-left corners may coincide with the node. A label touching the border verify one of the following statements: (1) its top-left corner touches an edge with a positive slope, (2) its bottom-left corner touches an edge with a negative slope, or (3) the rightmost node touches the left side of the label, and then the label has the same x-coordinate as N_{right}.

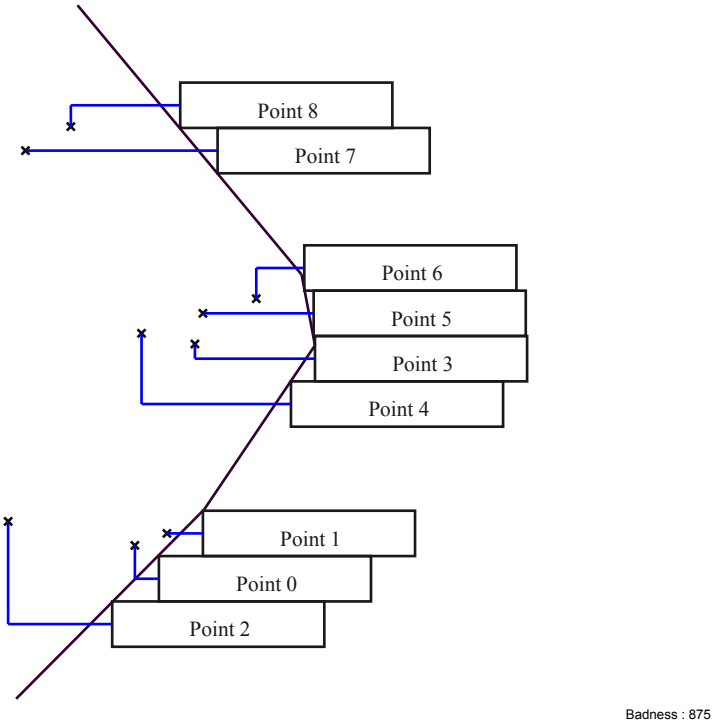


Figure 3.12: Example of a labeling for a convex border.

Furthermore, we saw in the previous section that a label has to be shifted when the x-coordinate of the point it is connected to is right to the x-coordinate the label would have if this label was touching the border. When an edge has a negative slope, the shifted label is connected by an upward leader. Similarly to the definition of the three kinds of downward leader, we define here three kinds of upward leaders:

1. *upward-straight* leaders: the label is not shifted, but is vertical.
2. *upward-shifted* leaders: the label is shifted and then vertical.
3. *upward-regular* leaders: the label is not vertical.

3.2.1 Leader Length Functions

In this part, we study the leader length functions and deduce properties with help computing a minimal cluster labeling.

Let s_j be the y-coordinate of a label. When the label touches the border, its x-coordinate t_j can be calculated with the border equation: $t_j(s_j) = \frac{s_j}{m_h} + a_h$, where E_h is the edge touching the label L_j. The previous section presented results for a convex border consisting in a single edge, i.e., when $k = 1$.

Consider a point P_i. We denote by $f_i(s_j)$ the length of the leader incident to P_i when the label it is connected to has y-coordinate s_j. The following lemma generalizes Lemma 5 to convex border composed of several edges.

Lemma 12. *The function $f_i(s_j)$ corresponding to a point P_i is piecewise linear and has no local non-global minimum, but is generally not convex. Moreover, the endpoints of the segments composing $f_i(s_j)$ correspond to straight leader or to y-coordinates $n_0, \ldots, n_k, n'_{right}$.*

Proof. Consider a point P_i connected to a label L_j, a convex border composed of k edges $E_1, \ldots, E_k$ and the $k+1$ corresponding nodes $N_0, \ldots, N_k$ so that two consecutive nodes N_{i-1} and N_i are the endpoints of the edge E_i.

Let n_{up} and n_{down} be the coordinates of the label L_j corresponding respectively to an upward-straight leader and a downward-straight leader. We prove here that (1) the minimum of the length function $f_i(s_j)$ of the leader $\ell(P_i, L_j)$ is attained when the leader is horizontal-straight, that (2) this function is piecewise linear and that (3) the endpoints of the segments composing $f_i(s_j)$ corresponds to the y-coordinates $y_i, n_{up}, n_{down}, n_1, \ldots, n_k$, and n'_{right}.

Let's position first the label at the y-coordinate y_i on an edge E_h, and move the label upward. We create then *events* corresponding to the y-coordinates $n_h, n_{h+1}, \ldots, n_{last}$ and n_{up} so that $\forall l > last$, $n_{up} < n_l$. In this order, the events are sorted by increasing values. We denote by e_l the l-th event.

Between the y-coordinates $s_j = y_i$ and $s_j = e_1$, the label stays on the edge E_h. When we move the label by a distance of $\varepsilon \leq e_1 - y_i$ upward, the leader length is increased by $(1+1/m_h)\cdot\varepsilon$. Since the slopes are supposed big, $|m_h| \geq 1$ and therefore $(1+1/m_h)\cdot\varepsilon \geq 0$, i.e., the leader length increases. When the y-coordinate of the label is between the two events e_l and e_{l+1}, the label stays on the same edge E_{h+l}. Thus, when moving the label by a distance of $\varepsilon \leq e_{l+1} - e_l$ upward between these two positions, the leader length increases by $(1 + 1/m_{h+l}) \cdot \varepsilon$. Finally, from the last event $e_l = n_{up}$, moving the label upward by a distance of ε increases the leader length by $\varepsilon \geq 0$ and the label becomes or is shifted. We conclude that the leader length $f_i(s_j)$ corresponding to a horizontal-straight leader is lower that the leader length corresponding to an upward leader. Moreover, the function $f_i(s_j)$ is piecewise linear for $s_j \geq y_i$ and the endpoints of the segments are y_i and the events $n_h, n_{h+1}, \ldots, n_{last}, n_{up}$ corresponding to an upward-straight leader, to a horizontal-straight leader and to several of the y-coordinates $n_1, \ldots, n_k, n'_{right}$.

We can prove analogously that the function $f_i(s_j)$ for $s_j < y_i$ is piecewise linear, and that the endpoints of the segments of the function correspond to a downward-straight leader, to a horizontal-straight leader and to several of the y-coordinates $n_1, \ldots, n_k, n'_{right}$.

Finally, we note that the leader length function presented in Figure 3.13 is not convex, for example between the y-coordinates y_i and n_{up}. □

We deduce from this lemma the following corollary, to compute a minimal labeling for a cluster of labels.

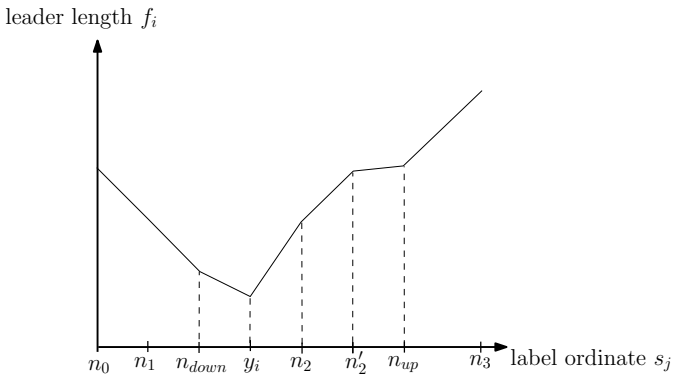
Corollary 2. *Given a set of labels that are grouped in the same cluster. There exists a minimal cluster labeling with either a horizontal-, or an upward- , or a downward-straight leader, or such that a left corner of a label touches a node of the convex border.*

Proof. We saw in the proof of Lemma 12 that the endpoints of the segments of the leader length function f_i for a point P_i correspond to a horizontal-straight leader, an upward-straight leader, a downward-straight leader or the overlay of a left corner of the label with a node of the border.

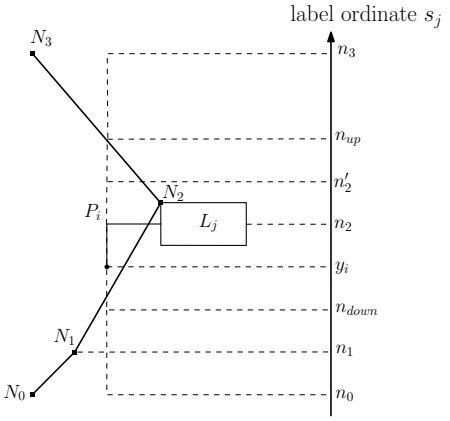
Given n points $P_0, \ldots, P_{n-1}$ and an order σ of the labels inside the cluster, the total leader length function depending on the y-coordinate of the bottommost label of the cluster is:

$$bad(s_0) = \sum_{i=0}^{n-1} \big(f_i(s_o + \sigma(i) \cdot h)\big)$$

Therefore, the total leader length function is a piecewise linear function, and one of its minimums is attained at an endpoint of a segment of one of the functions f_i. □



(a) Leader length function



(b) Border considered

Figure 3.13: length function of a leader given the border equation and the coordinates of the point it is connected to.

3.2.2 Minimal Cluster Labeling using a Matching Algorithm

In this part, we are given n points $P_0, \ldots, P_{n-1}$. Each point P_i has to be connected to one of the n labels $L_0, \ldots, L_{n-1}$. We suppose here that the n labels are regrouped into a single cluster. The objective is to compute a position of this cluster and an order σ of the labels depending on the point they are connected to, such that the point P_i is connected to the label $L_{\sigma(i)}$.

When the border consists of a single edge, we saw that the Algorithm of Vaidya [Vai89] computes the optimal order of labels in $O(n^2 \cdot (\log n)^3)$ time, given fixed positions for the labels.

For the case of a convex border with k edges, we know from Corollary 2 that one of the minimal labelings fulfills at least one of the following statements:

- there is a horizontal-straight leader,
- there is a downward-straight leader,
- there is an upward-straight leader,
- a corner of one of the labels touches a border node.

These statements define $O(k \cdot n^2)$ possible cluster positions. Thus, Algorithm 3 can easily be generalized for convex borders. Thus, we conclude:

Theorem 9. *There exists an $O(n^4(\log n)^3)$ time algorithm that computes an minimal cluster labeling.*

Algorithm 9 computes in $O(k \cdot n^4(\log n)^3)$ time algorithm that calls $O(k \cdot n^2)$ times the algorithm of Vaidya [Vai89] in order to compute a minimal cluster labeling:

Algorithm 9: Minimal Cluster WAB-Labeling for a convex border with big slopes.

Data: $P_0, \ldots, P_{n-1}$.
Result: *cluster_position*, σ for a minimal cluster labeling.
List_Positions := {};
foreach $j \in \{0 \ldots n-1\}$ **do**
  foreach $i \in \{0 \ldots n-1\}$ **do**
    List_Positions $\cup = \{cluster\_position \mid \ell(P_i, P_j)$ is horizontal-straight$\}$;
    List_Positions $\cup = \{cluster\_position \mid \ell(P_i, P_j)$ is upward-straight$\}$;
    List_Positions $\cup = \{cluster\_position \mid \ell(P_i, P_j)$ is downward-straight$\}$;
  foreach $i \in \{0, \ldots k\}$ **do**
    List_Positions $\cup = \{cluster\_position \mid$ a corner of L_j touches $N_i\}$;
Remove redundancy of cluster positions in *List_Positions*;
best_badness := upper bound of the minimum badness;
best_position := 0;
*best_*σ := $Y - order$;
foreach *cluster_position* $\in$ *List_Positions* **do**
  (*badness*, σ) :=OptimalMatching(*cluster_position*);
  if *badness* $<$ *best_badness* **then**
    best_badness := *badness*;
    best_position := *cluster_position* *best_*σ := σ;
return *cluster_position*, σ

3.2.3 Minimal Cluster Labeling using a Sweep-line Algorithm

In this part, we suppose that every label is grouped into the same cluster, and we look for a minimal cluster labeling, that is, an order of the labels and a position of the whole cluster which minimizes the total leader length. We consider that the slope of each of the k edges is big and we allow crossings between two leaders and between a leader and a label.

In the previous section, we saw two algorithms. Algorithm 4 computes a cluster labeling minimizing a badness bad' given an order σ of the labels. We saw that, when we set σ to be the Y-order, the cluster labeling computed is minimal if it contains no shifted label. Algorithm 5 is a sweep-line algorithm computing a minimal cluster labeling for a given order of labels. We proved with Lemma 8, due to the convexity property of the leader length functions, that the cluster labeling computed by the algorithm is a minimal cluster labeling if it contains no vertical leaders, that is no downward-straight or -shifted leader.

Corollary 2 states that there exists a minimal cluster-labeling containing one of the followings:

- a horizontal-straight leader
- an upward-straight leader
- a downward-straight leader
- a label whose top-left corner or bottom-left corner overlay a node of the border.

Therefore, we can easily generalize the sweep-line algorithm used in Algorithm 5 to compute a minimal cluster labeling for a given order of the labels, by taking into account the positions of the cluster for which an upward-shifted leader get on an edge and the positions

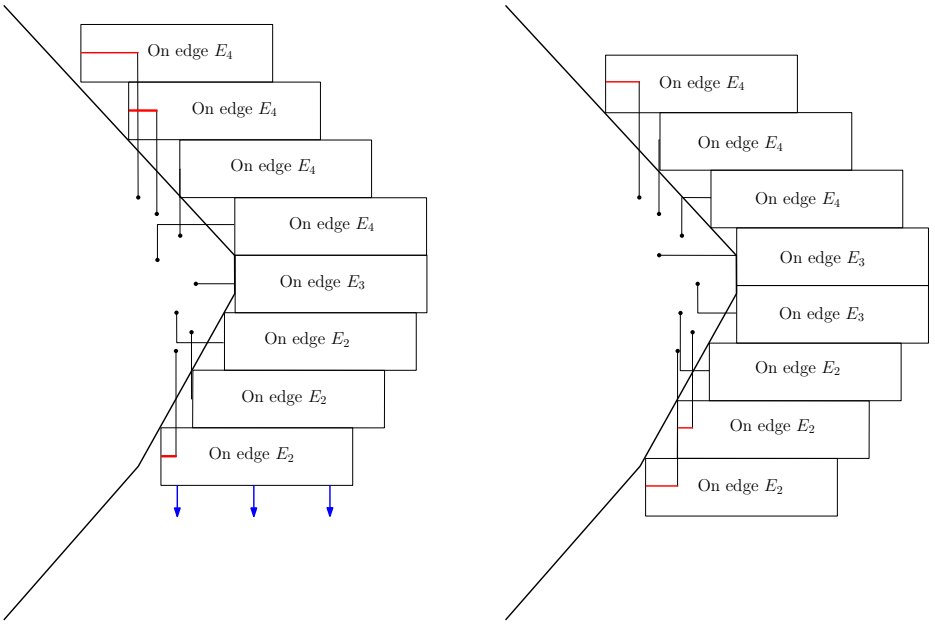
where a label change the edge it is on. However, we do not describe here precisely how to do it, because the leader length functions are not convex anymore, which makes it more difficult to find a sufficient condition so that the minimal cluster labeling among the cluster labelings using the Y-order is a minimal cluster labeling.

In this part, we create a sweep-line algorithm similar to Algorithm 5 generalizing Algorithm 4, that is, computing a labeling minimizing the same badness function bad' as in Algorithm 4. Then, we prove that, if the cluster labeling computed using the Y-order contains no shifted label, then this cluster labeling is minimal. Since the condition of minimality of the labeling is only granted with the Y-order, we suppose here that the labels are sorted with respect to the Y-order.

We assume now that we have n points $P_0, \ldots, P_{n-1}$ and the equation Eq of a convex border consisting of k edges $E_1, \ldots, E_k$. We denote by $L_i : (t_i, s_i)$ the coordinates of the label connected to the point $P_i : (x_i, y_i)$. Let t'_i be the x-coordinate of the label L_i when L_i touches the border. The sweep-line algorithm we describe here is based on the proof of Lemma 6. Therefore, we look for a cluster-labeling $\mathcal{L}$ minimizing the following badness function:

$$\text{bad}'(\mathcal{L}) = \sum_{i=1}^{n} |y_i - s_i| + (x_i - t'_i)$$

We recall that the badness function bad$'$ depends on t'_i instead of t_i, and then everything happens as if the labels are on the border, and instead of shifted labels, we have leader with negative horizontal leader length. see Figure 3.14. In particular, we do not consider the positions of the cluster corresponding to an upward-straight or downward-straight leader.



(a) Initial cluster position: a straight leader and a corner on a border node.

(b) Next cluster position: s straight leader.

Figure 3.14: Cluster moving downward by a distance of ε. The red segments represent negative leader length for the badness bad$'$. The blue arrows represents the shift distance ε

A cluster position minimizing this badness function bad$'$ can be computed with a sweep-line algorithm. We first compute the badness bad$'$ corresponding of the topmost cluster position, which may not correspond to a WAB-labeling. Then, we iteratively compute the

badness bad′ from top to bottom for the cluster positions.

Algorithm 10: Possibly Minimal Cluster WAB-Labeling using the Y-order

Data: points $P_0, \ldots, P_n$
Result: A possibly minimal cluster labeling
$Y' = \{\}$;
foreach $i \in \{0 \ldots n-1\}$ **do**
  add $(y_i - i \cdot h$,HS) to Y';
  foreach $j \in \{0 \ldots k\}$ **do**
    add $(n_{i,j} - i \cdot h$,Nj) to Y';
Sort Y' in decreasing order;
new_s_0 := cluster position corresponding to the first element of Y';
new_ll := badness bad′ for position new_s_0;
n_s := number of horizontal-straight leaders for position new_s_0;
n_u := number of upward leaders for position new_s_0;
foreach $j \in \{0 \ldots k\}$ **do**
  $E[j]$:= number of labels touching edge E_j for position new_s_0;
foreach *cluster position in* Y' **do**
  previous_s_0 := new_s_0;
  previous_ll:= new_ll;
  new_s_0 := y-coordinate of the cluster position;
  $n_u := n_u - n_s$;
  n_s := number of element corresponding to the next cluster position with second field with value HS;
  foreach *element with second field Nj* **do**
    $E[j] := E[j] - 1$;
    $E[j-1] := E[j-1] + 1$;
  new_ll := previous_ll + (previous_s_0 - new_s_0) $\cdot\big(n - 2 \cdot n_u - \sum_j \frac{E[j]}{m_j}\big)$;
  update the best badness and the associated s_0 value.;
return the best labeling;

Lemma 13. *If the labeling provided by Algorithm 10 contains no shifted labels, then this is a minimal cluster labeling.*

Proof. We prove this lemma exactly the same way as we proved Lemma 13. We redefine here the projection operation proj($\mathcal{L}$) which consist of moving the shifted label of a labeling $\mathcal{L}$ back to the border.

We denote here by bad($\mathcal{L}$) the total leader length of a labeling $\mathcal{L}$ and bad′($\mathcal{L}$) = F_1(proj($\mathcal{L}$)) the following virtual badness function, depending on the order of the labels $\sigma_{\mathcal{L}}$ and the x-coordinate t'_j of the labels in the projection proj($calL$) of the labeling $\mathcal{L}$:

$$\begin{array}{rcl} \text{bad}'(\mathcal{L}) & = & \sum_{\ell(P_i, L_{\sigma_{\mathcal{L}}(i)})} \Big(|s_{\sigma_{\mathcal{L}}(i)} - y_i| + (t_{\sigma'_{\mathcal{L}}(i)} - x_i) \Big) \\ & \leq & \sum_{\ell(P_i, L_{\sigma_{\mathcal{L}}(i)})} \Big(|s_{\sigma_{\mathcal{L}}(i)} - y_i| + |t_{\sigma_{\mathcal{L}}(i)} - x_i| \Big) \\ \text{bad}'(\mathcal{L}) & \leq & \text{bad}(\mathcal{L}) \end{array}$$

Suppose that the labeling $\mathcal{L}^*$ computed by Algorithm 10 contains no shifted label. Therefore, we have:

$$\text{bad}(\mathcal{L}^*) = \text{bad}'(\mathcal{L}^*) \leq \text{bad}'(\mathcal{L}),\ \forall \mathcal{L} \text{ using the } Y\text{-order} \tag{1}$$

Consider now a labeling $\mathcal{L}_2$. The value of its badness bad' is lower than its total leader length:

$$\text{bad}(\mathcal{L}_2) \geq \text{bad}'(\mathcal{L}_2) \tag{2}$$

Switch the labels of the cluster to create a labeling $\mathcal{L}_3$ with the same position as $\mathcal{L}_2$, but using the Y-order. The labeling $\mathcal{L}_3$ has a value b_3 for the badness function bad'. For the badness bad', every cluster labeling with the same position for the cluster has the same horizontal leader length $\sum(t_i) - \sum(x_i)$. Given a set of n points and n slots for the labels, Lemma 2 in Section 2.4 states that the vertical leader length is minimized with the Y-order. We deduce:

$$\text{bad}'(\mathcal{L}_2) \geq \text{bad}'(\mathcal{L}_3) \tag{3}$$

Because of Equation (1), we have

$$\text{bad}'(\mathcal{L}_3) \geq \text{bad}'(\mathcal{L}^*) \tag{4}$$

From (1), (2), (3) and (4), we conclude:

$$\text{bad}(\mathcal{L}_2) \geq \text{bad}'(\mathcal{L}_2) \geq \text{bad}'(\mathcal{L}_3) \geq \text{bad}'(\mathcal{L}^*) = \text{bad}(\mathcal{L}^*)$$

We conclude that $\mathcal{L}^*$ is a minimal cluster labeling. □

Theorem 10. *Algorithm 10 computes a minimal cluster labeling for the given order of labels in $O(k \cdot n \log(k \cdot n))$ time.*

Proof. The correctness follows from Lemma 13. We need only to prove the time complexity. The algorithm consists of sorting a list of $O(n \cdot k)$ elements, which require $O(k \cdot n \log(k \cdot n))$ time, and then sweeping its elements. For each element of the list, The edge touching the label can be computed with a binary search algorithm in $O(\log k)$ time. Thus, the sweep-line takes $O(k \cdot n \log(k))$ time. □

3.2.4 Removing the crossings

In this part, we suppose that we have a minimal labeling for a set of n points. The objective is to remove the leader crossings and the leader-label crossings. When the border is composed of a single edge, we saw that Algorithm 2 removes the leader crossings whereas Algorithm 6 removes the leader-label crossings. The two algorithms can be run in an arbitrary order.

The equation of the border has no direct influence on Algorithm 6, the only assumption made is the possibility to have downward-shifted leaders. For a convex border, we also have to take into account the presence of upward-shifted leaders.

With the same reasoning as in the previous section, we create the following algorithm to

remove the leader-label crossings:

Algorithm 11: Algorithm Removing the Leader-Label Crossings, for Convex Boundary

Data: σ $L_0, \ldots, L_{n-1}$) for minimal labeling
Result: Removes leader-label crossings.
// removing crossing involving downward-shifted leaders ;
foreach j *from* $n-1$ *to* 0 **do**
  $i = \sigma^{-1}(j)$.;
  if $\ell(P_i, L_j)$ *has type po-leader and involved in a downward crossing* **then**
    $\ell(P_{\sigma^{-1}(k)}, L_k)$:= downward leader with the bottommost crossing with $\ell(P_i, L_j)$.;
    Switch the position L_j and L_k.;
// removing crossing involving upward-shifted leaders ;
foreach j *from* 0 *to* $n-1$ **do**
  $i = \sigma^{-1}(j)$.;
  if $x_i = t_j$ **then**
    $k := argmax\{t_k \mid (k < j)\&(t_k = x_{\sigma^{-1}(k)})\}$.;
    Switch the position of L_j and L_k.;

Theorem 11. *Given a minimal labeling, Algorithm 11 removes in $O(n^2)$ time the leader-label crossings and Algorithm 2 removes in $O(n^2)$ time the leader crossings.*

Proof. Lemma 10 proves that the first iteration removes the leader-label crossings involving a downward-shifted leader. Since the problem is symmetrical, we can prove with an analogous reasoning that the second iteration removes the leader-label crossings involving an upward-shifted leader.

since the two iterations are similar, they both have the same complexity. Theorem 11 states that the first iteration takes $O(n^2)$ time. Therefore Algorithm 11 also takes $O(n^2)$. □

3.2.5 Computing an Optimal Labeling

In the previous parts, we saw two algorithms to compute a minimal cluster labeling, and an algorithm to remove the crossings from a minimal labeling. In this part, we regroup the results and deduce an algorithm to compute an optimal labeling. The matching algorithm presented previously always compute a minimal cluster labeling, but takes $O(n^4(\log n)^3)$ times. The sweep-line algorithm only takes $O(n{\cdot}k\log(n{\cdot}k))$ time, but the computed cluster labeling is only verified to be optimal when it contains no shifted label. It is reasonable to think that, in most cases, the points will be placed so that the labeling computed by the sweep-line algorithm contains no shifted label. We create then an algorithm that (1) first compute a labeling with the sweep-lime algorithm, and (2) only call the algorithm using a matching when the first computed labeling contains a shifted label. The following algorithm computes a minimal cluster labeling with a best case time complexity of $O(n \cdot$

$k \log(n \cdot k)$) and with a worst case complexity of $O(n^4(\log n)^3)$:

Algorithm 12: Minimal Cluster WAB-Labeling for Convex Border with big slopes

Data: Points $P_0, \ldots, P_{n-1}$
Result: Labels $L_0, \ldots, L_{n-1}$ and order σ for a minimal cluster labeling.
Use Algorithm 10 with $\sigma := Y - order$ to compute a position of the labels $L_0, \ldots, L_{n-1}$;
if $\exists j \mid L_j$ *shifted* **then** return the labeling computed by 9;
else return $Y - order$, $L_0, \ldots, L_{n-1}$;

In Section 3.1, Lemma 11 states that the Upward-downward algorithm allows to compute optimal labeling when the leader length function are convex. Unfortunately, it is not the case when the border is composed of several edges. The total leader length function of a cluster C may have two different optimal positions c_1 and c_2 so that positioning C at the position c_2 creates an intersection with another cluster. The minimal labeling is then computed by positioning C at the position c_1, whereas fusing the two cluster would result to a greater badness.

Therefore, when two clusters C_1 and C_2 intersect, we have to test the minimal cluster labelings in the following cases:

- ($Test$ 1) we merge the clusters C_1 and C_2.
- ($Test$ 2) we do not merge the cluster and place them at a local minimum cluster labeling so that C_1 and C_2 do not overlap.

Moreover, moving a cluster C_1 downward may create a new intersection between this cluster and the cluster C_0 below it. Thus, we should repeat these two tests for each intersection created by moving a cluster downward or recomputing a minimal cluster labeling after fusing two clusters. This leads to an algorithm with an exponential time complexity.

However, this problem can only happen when both positions of the cluster touch several edges. If the minimal cluster labeling touches a single edge E_i, then everything happens locally for the cluster as if the border only consists in E_i, and the badness function of a cluster only contain a global minimum. Then, if two intersecting clusters contain labels on a single edge, we have to merge them.

Furthermore, suppose, in ($Test$ 2), that the cluster C_2 is moved upward and the cluster C_1 is moved downward. When considering the intersection between C_1 and the cluster C_0 below it, we study the two following cases. Either, the clusters are merged, or the cluster C_0 moves downward, and C_1 stays at the same place, because moving the cluster C_1 back upward corresponds to a case already tested. We only test what happens when the cluster C_0 moves downward when it touches several edges. Moreover, the set of edges two non overlapping clusters are on can only have two edges in common. Therefore, if the cluster C_1 is on the edges $E_i, \ldots, E_j$, then no label of the cluster C_0 can be on the edge E_{i+1}. Therefore, the number of times we have to test moving a cluster downward depend on the number of edges $O(k)$. We conclude that the number of minimal cluster labeling we have to compute is $O(n \cdot 2^k)$.

This remark leads to the following adapted version of the Upward-downward algorithm:

At each step of the n steps of the iteration, the algorithm computes at most 2^k cluster labeling, therefore the complexity of Algorithm 13 is $O(n \cdot 2^k \cdot \text{ClusterPos}(n))$, where $\text{ClusterPos}(n)$ is the time complexity to compute a minimal cluster labeling for a cluster with n labels and where k is the number of border edges. Unfortunately, no more efficient method to compute a minimal labeling has been found. Moreover, it is not easy to prove the

Algorithm 13: Upward-Downward Algorithm adapted for a convex border with several edges.

Data: Points $P_0, \ldots, P_{n-1}$, border equation, algorithm to compute minimum cluster labeling.
Result: Computes a minimal labeling.
for $i = 0$ *to* $n-1$ **do**
  Create a cluster with the label L_i connected to P_i;
  while *Cluster with* L_i *intersect a cluster* **do**
    if *the two cluster touches both a single edges* **then**
      Merge the two clusters, and compute a minimal cluster labeling for the resulting cluster;
    else
      Test the 2^k different possibilities of cluster merging, and keep the possibility with the minimal badness;
return the resulting labeling;

correctness of this algorithm. In Section 4.3, we will present in another model of labeling a contraint of *local optimality*. With this contraint, we will prove that the Upward-downward algorithm induces an optimal labeling.

The following algorithm regroups different algorithms to compute an optimal labeling in $O(n \cdot 2^k \cdot \text{ClusterPos}(n))$:

Algorithm 14: Optimal WAB-Labeling for Convex Border with big slopes

Data: Points $P_0, \ldots, P_{n-1}$
Result: Labels $L_0, \ldots, L_{n-1}$ and order σ for an optimal labeling.
Compute a minimal cluster labeling with an Upward-downward Algorithm, given in input Algorithm 12 providing a minimal cluster labelings;
Call Algorithm 2 to remove leader crossings;
Call Algorithm 11 to remove leader-label crossings;
return the labeling;

3.2.6 Integer Linear Programming

The integer linear program we modeled for a single edge can not be easily generalized for several edges. Indeed, a constraint defining the horizontal length was:

$$t(s_j) - \sum_i (\sigma(i \cdot n + j) \cdot x_i) \leq llx[j]$$

Where llx, σ and s_j are variables, and j and x_i are fixed values. We had also $t(s_j) = a + s_j/m$, where a and m are constants. However, for a convex border, the new value of $t(s_j)$ is $t(s_j) = \min_i\{a_i + s_j/m_i\}$, where a_i and m_i are constants. However, in an integer linear program, it is not easy to create a constraint stating that a variable has to be bigger that the minimum of non-constant values. Since the minimum here depends on the variables $t(s_j)$, we cannot compute this minimum in a Integer Linear Program, and adding a new variable would produce a multiplication between two variables. The integer program created is then quadratic.

We will see in Section 4.2 an integer linear program to compute a minimal labeling for a convex border.

3.2.7 Conclusion

In this section, we extended the algorithm developed for a single-edge border in the case of a convex border composed of k edges. In both cases, we assumed that the slope of the edges are big, i.e., each equation of the x-coordinate of a label $L : (t, s)$ on an edge E_i of the border given its y-coordinate has the form $t(s) = a_i + s/m_i$ with $|m_i| > 1$.

A new difficulty appears when the border consists of more than a single edge, namely the non-convexity of the leader length functions. Because of it, we have to rethink and adapt the Upward-downward algorithm to compute the regrouping of the labels into clusters. As a consequence, the running time of the algorithm is increased by an exponential factor in the number of edges k and no proof of correctness has been found when a cluster of labels touches several edges. However, we will see a strong constraint in a later section with which the Upward-downward algorithm computes an optimal labeling.

Algorithm 14 allows to compute optimal labelings for different shapes of the boundary, and thereby reducing the white spaces between the border and the surface delimiting the map.

An interesting research topic would be to study two-sided dynamic labelings, which has not been completely studied, even for vertical borders.

4. Alternative Labeling Models

In Chapter 3, we studied a first model to compute optimal labelings. In this chapter, we present three further models. The first section describes an alternative model to the weakly-aligned boundary labelings where we require that every label touches the boundary. Then, we use integer linear programming to solve the convex boundary problem in a general case. This integer linear program will resolve two difficulties that occurs in the WAB-labeling model. First, an ILP instance can be solved by known algorithms. Therefore, we overcome the difficulty encountered in the proof of the Adapted upward-downward algorithm. Moreover, the ILP allows to generalize our model to convex borders containing edges with small slopes $|m| \leq 1$ and to two-sided boundary labelings. Finally, we will present an alternative approach of the boundary labeling problem by requiring that labels in a same cluster are vertically aligned. For this last model, we will develop algorithm computing optimal labelings with the same time complexity as the algorithm presented in Section 2.4.

4.1 Strictly-Aligned Boundary Labeling

In this section, we study an alternative positioning of the labels to the WAB-labeling model used in Chapter 3. When a cluster contains several shifted labels, it may be difficult to follow the leaders with the eyes, since the leaders corresponding to shifted leaders are then parallel lines close to each others. Figure 4.1 shows an example of cluster labeling containing several downward-shifted leaders.

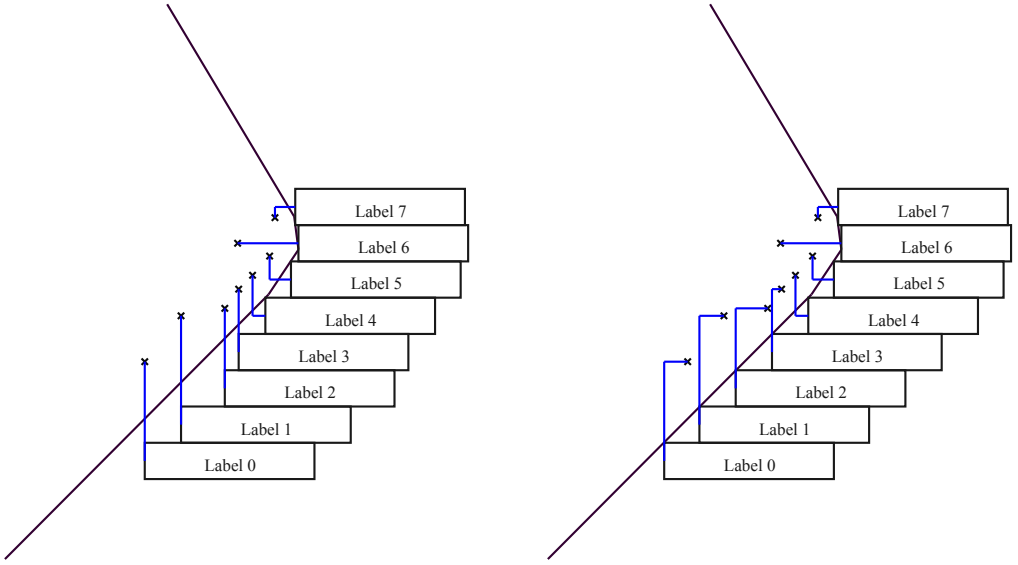
On the other hand, forbidding shifted labels in the labelings may results in unfeasible labelings, see Figure 4.2 for an example.

An alternative is to require labels to be on the border, and allowing the use of op-leaders to allow points to be connected to labels to the left of them. We call this model of labeling *strictly-boundary aligned labeling* or *SAB-labeling*.

In this section, we assume that we have the same input as in section 3.2. Our objective is to produce a crossing-free SAB-labeling with minimal leader length.

4.1.1 General Results

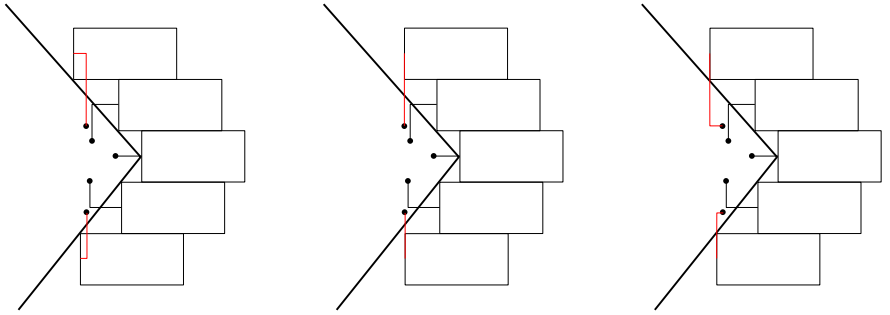
We first study general results such as the leader length functions and the upward-downward algorithm.



(a) Labeling with shifted labels

(b) Labeling with op-leaders

Figure 4.1: Illustration of a cluster with several neighborings shifted labels. Figure (a) shows a WAB-labeling where several labels are shifted. In Figure (b), the shifted labels have been replaced by labels that are touching the border and which are connected by op-leaders.



(a) Infeasible Labeling

(b) Labeling allowing shifted labels

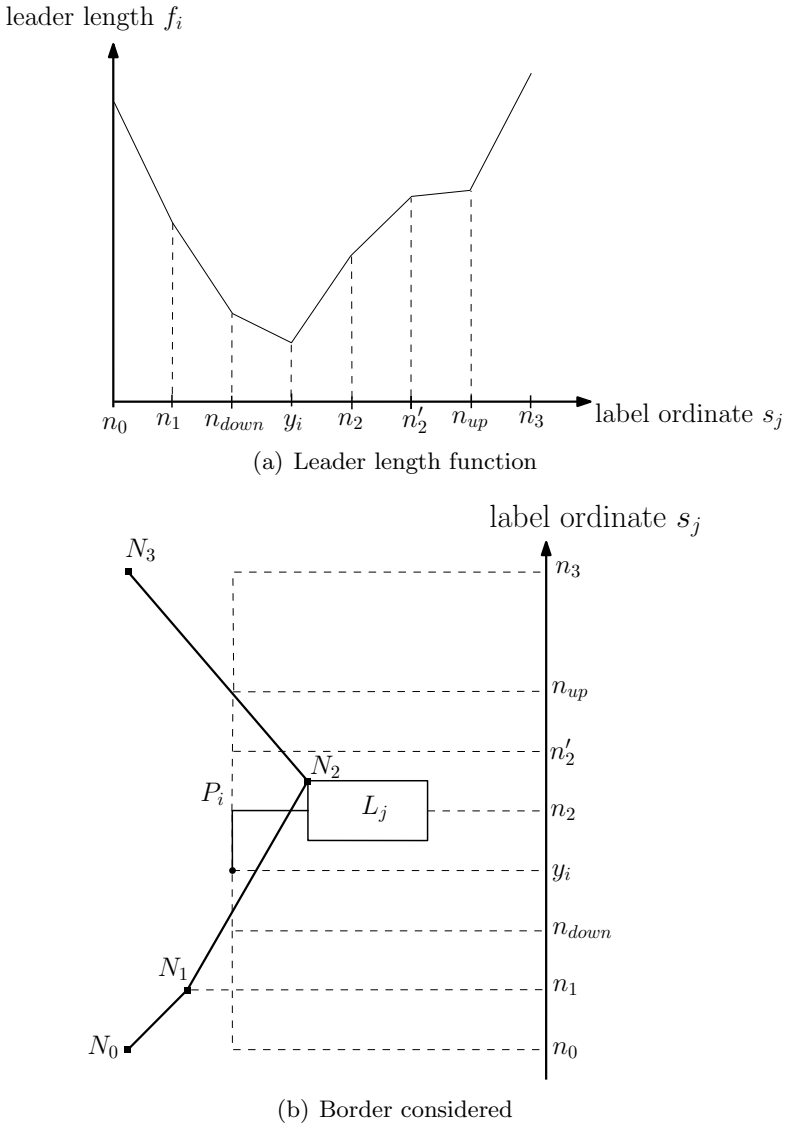
(c) Labeling allowing op-leaders

Figure 4.2: Different kinds of labelings the same input data. If we forbid shifted labels without allowing a new type of leader, the labeling is infeasible: In figure (a), we cannot place the cluster without having either the topmost or the bottommost label misplaced. In (b) and (c), these label are well placed because we allowed shifted labels of op-leaders.

Let P_i be a point in the map, and $f_i(s)$ be the length of the leader connected to P_i when the label it is connected to has y-coordinate s. We assume that the boundary consists of k edges $E_1, \ldots, E_k$ connected by the nodes $N_0, \ldots, N_k$. Let N_{right} be the rightmost node of the border. We denote by n_j the y-coordinate of a label so that the node N_j coincides with one of the corners of this labels. For $N_j = N_{right}$, we define a further y-coordinate n'_{right}, because in this case, both the top-left and the bottom-left corners may coincide with the node.

Lemma 14. *The function $f_i(s)$ corresponding to any point P_i is piecewise linear and has no local non-global minimum. Moreover, this function is convex when the border consists of a single edge.*

Proof. This proof follows exactly the same argumentation as the proof of Lemma 12. When the label is not shifted, the leader length are exactly the same for a WAB-labeling and for a strict labeling. Between the two labeling models, $f_i(s)$ only change when the label is 'shifted' in a WAB-labeling. A shifted label in a WAB-labeling corresponds to an op-lea1der in a SAB-labeling, and the leader length of an op-leader contains a vertical part which increases as the label is moved away from the label. Moreover, in WAB-labeling, some border nodes have no influence on the leader length because its y-coordinate corresponds to a shifted label, like node N_1 in Figure 4.3. For a SAB-labeling, these nodes change the slope of a segment of the function since it modifies the horizontal leader length. But in general, it is easy to understand that the further away we move a label from the optimal position y_i, the bigger the length of the leader gets. □



(a) Leader length function

(b) Border considered

Figure 4.3: Length function of a leader for a given the border equation and the coordinates of the point it is connected to. The border used and the position of the point P_i are exactly the sames as in Figure 3.13.

As in the previous sections, we can reduce the number of positions which are necessary to consider when looking for a minimal cluster labeling.

Corollary 3. *Given a set of labels that belong to the same cluster. There exists a minimal cluster SAB-labeling with a horizontal-, upward- or downward-straight leader, or so that one of the labels has one of the y-coordinates $n_0, \ldots, n_k$, or n'_{right}. The number of possible positions to the cluster is then reduced to $O(k \cdot n^2)$.*

Furthermore, because the properties of the leader length functions in this section and in the previous one are the same, the adapted upward-downward algorithm explained in Section 3.2 also create clusters corresponding to a minimal SAB-labeling:

Lemma 15. *Algorithm 13 creates a clustering for a minimal SAB-labeling, when it is given as input an algorithm to compute a minimal SAB-labeling for a single cluster of labels.*

4.1.2 Minimal Cluster Labeling using a Matching Algorithm

For WAB-labelings, we solved the minimal cluster labeling problem by computing several minimum weighted matchings. We do here exactly the same. Given a fixed position for a cluster, i.e., the y-coordinates of the labels in the cluster, we create a complete bipartite graph between the n vertices representing the points and the n vertices representing the labels. The cost of an edge is equal to the length of the leader that would connect the corresponding point and label.

The only difference between the matching created in Section 3.2 and the matching here is the cost of the op-leaders. Thus, this problem is exactly the same as in Section 3.2.

Algorithm 15: Minimal Cluster SAB-Labeling for a Convex Border with big slopes.

Data: $P_0, \ldots, P_{n-1}$
Result: *cluster_position*, σ for a minimal cluster SAB-labeling.
List_Positions := {};
for $j = 0$ *to* $n-1$ **do**
  for $i = 0$ *to* $n-1$ **do**
    List_Positions $\cup = \{$*cluster_position* $\mid \ell(P_i, P_j)$ is horizontal-straight$\}$;
    List_Positions $\cup = \{$*cluster_position* $\mid \ell(P_i, P_j)$ is upward-straight$\}$;
    List_Positions $\cup = \{$*cluster_position* $\mid \ell(P_i, P_j)$ is downward-straight$\}$;
  for $i = 0$ *to* k **do**
    List_Positions $\cup = \{$*cluster_position* $\mid$ a corner of L_j touches $N_i\}$;
Removes redundancy of cluster positions in *List_Positions*;
best_badness := upper bound of the minimum badness;
best_position := 0;
*best_*σ := Y-order ;
for *cluster_position* $\in$ *List_Positions* **do**
  (*badness*, σ) :=OptimalMatching(*cluster_position*);
  if *badness* $<$ *best_badness* **then**
    best_badness := *badness*;
    best_position := *cluster_position*;
    *best_*σ := σ;
return *cluster_position*, σ

Theorem 12. *Algorithm 15 computes in $O(k \cdot n^4 (\log n)^3)$ time a minimal cluster SAB-labeling.*

Proof. The correctness of the theorem follows from Lemma 14 and the correctness of the Vaidya algorithm [Vai89] used to compute a minimum weighted matching. Algorithm 15

computes $O(k \cdot n^2)$ matchings via the algorithm of Vaidya, where a single computation of a matching requires $O(n^2 \cdot (\log n)^3)$ time. Therefore, Algorithm 15 takes $O(k \cdot n^4 (\log n)^3)$ time. □

4.1.3 Minimal Cluster Labeling using a Sweep-line Algorithm

In this part, we are given a set of n points and a cluster of n labels. In Section 3.2, Algorithm 10 computes a cluster WAB-labeling using the Y-order, which is minimal if it contains no shifted labels. The objective of this part is to adapt this approach to the SAB-labelings. The interesting point of this sweep-line algorithm is that the computed labeling is optimal if it does not contain any no shifted labels, which corresponds to containing no op-leader in the model considered here. Moreover, a minimal cluster labeling only depends on the leader length functions. Just as in Section 3.2, we define a virtual badness function bad' and compute a labeling which minimizes this badness function. Algorithm 10 computes a minimal labeling $\mathcal{L}$ for the following badness function bad':

$$bad'(\mathcal{L}) = \sum_i \Big(|y_i - s_i| + (t_i - x_i) \Big)$$

We saw then that, given a WAB-labeling $\mathcal{L}$, the badness function bad' is lower or equal to the total leader length function bad, and the functions are the same if there are no shifted labels. For this reason, we proved that if the labeling computed by Algorithm 10 using the Y-order contains no shifted label, this labeling is then a minimal cluster labeling.

It is easy to check that, given an SAB-labeling $\mathcal{L}'$, this badness function bad' is also lower or equal to the total leader length function, and are the same if there are no op-leader. Therefore, with the same approach as in Section 3.2, Lemma 16 states that the cluster labeling computed by Algorithm 10 using the Y-order is also minimal in the case of the SAB-labelings if it contains no op-leader.

Lemma 16. *If the labeling provided by Algorithm 10 with the Y-order contains no op-leader for the Y-order, then it is a minimal cluster SAB-labeling.*

Proof. In this proof, we only recapitulate the main steps as in the proof of Lemma 13. Let $\text{bad}(\mathcal{L})$ be the total leader length in a labeling $\mathcal{L}$ Let $\mathcal{L}^*$ be the labeling computed by Algorithm 10. Let $\mathcal{L}_2$ be another labeling. Consider the labeling $\mathcal{L}_3$ at the same position as $\mathcal{L}_2$, but using the Y-order. The part of the badness bad' corresponding to the horizontal part of the leader remains unchanged, but the part corresponding to the vertical part cannot increase. Therefore, $\text{bad}'(\mathcal{L}_2) \geq \text{bad}(\mathcal{L}_3)$. Moreover, $\mathcal{L}_3$ and $\mathcal{L}^*$ use both the Y-order. By construction, the badness $\text{bad}'(\mathcal{L}_3)$ is bigger than $\text{bad}'(\mathcal{L}^*)$. We deduce:

$$\text{bad}(\mathcal{L}_2) \geq \text{bad}'(\mathcal{L}_2) \geq \text{bad}'(\mathcal{L}_3) \geq \text{bad}'(\mathcal{L}^*)$$

If the labeling $\mathcal{L}^*$ computed by Algorithm 10 contains no op-leader, then it badness $\text{bad}'(\mathcal{L}^*)$ is equal to its total leader length $\text{bad}(\mathcal{L}^*)$. Finally, we have

$$\text{bad}(\mathcal{L}_2) \geq \text{bad}(\mathcal{L}^*)$$

We conclude that $\mathcal{L}^*$ is a minimal cluster labeling. □

4.1.4 Crossing Removal

In this part, we generate a crossing-free minimal labeling from a minimal labeling. We assume that we have a placement of the labels and know which point is connected to which leader. The goal is to remove the leader crossings.

Lemma 17. *For a given a minimal cluster SAB-labeling the following holds:*

- *no op-leader can have a crossing with a non-vertical po-leader.*
- *no vertical leader can have a crossing with an op-leader as well as with a po-leader.*
- *There can be no leader-label crossings.*

Proof. We first prove that there can be no crossing between a po-leader and an op-leader in a minimal SAB-labeling. Suppose that a po-leader $\ell(P_i, L_{\sigma(i)})$ crosses an op-leader $\ell(P_j, L_{\sigma(j)})$. Without loss of generality, we suppose that the po-leader is downward. Several possibilities have to be considered, depending on the position of the crossing an whether the op-leader is upward or downward.

- If the op-leader is downward, and its horizontal segment crosses the po-leader (see Figure 4.4(a)). Because of the crossing, we have:

$$s_{\sigma(j)} < s_{\sigma(i)} < y_j < y_i \tag{1}$$

and

$$t_{\sigma(j)} < x_i < x_j, t_{\sigma(i)} \tag{2}$$

The length bad_1 of these two leaders is then:

$$bad_1 = (y_i - s_{\sigma(i)}) + (y_j - s_{\sigma(j)}) + (t_{\sigma(i)} - x_i) + (x_j - t_{\sigma(j)}) \tag{3}$$

Suppose now that we switch the two labels $L_{\sigma(i)}$ and $L_{\sigma(j)}$. The resulting labeling has a new badness bad_2:

$$bad_2 = (y_i - s_{\sigma(j)}) + (y_j - s_{\sigma(i)}) + (x_i - t_{\sigma(j)}) + |x_j - t_{\sigma(i)}| \tag{4}$$

Because of (1), the vertical leader length of the two labelings is the same:

$$(y_i - s_{\sigma(i)}) + (y_j - s_{\sigma(j)}) = (y_i - s_{\sigma(j)}) + (y_j - s_{\sigma(i)}) \tag{5}$$

From (2), we deduce:

$$x_j - t_{\sigma(j)} = (x_j - x_i) + (x_i - t_{\sigma(j)}) \tag{6}$$

$$(t_{\sigma(i)} - x_i) + (x_j - x_i) = 2 \cdot \big(\min\{x_j, t_{\sigma(i)}\} - x_i\big) + |x_j - t_{\sigma(i)}| \tag{7}$$

By introducing (5), (6), (7) into (3), we get:

$$\begin{array}{rcl} bad_1 & = & (y_i - s_{\sigma(j)}) + (y_j - s_{\sigma(i)}) + (t_{\sigma(i)} - x_i) + (x_j - t_{\sigma(j)}) \\ & = & (y_i - s_{\sigma(j)}) + (y_j - s_{\sigma(i)}) + (t_{\sigma(i)} - x_i) + (x_j - x_i) + (x_i - t_{\sigma(j)}) \\ & = & (y_i - s_{\sigma(j)}) + (y_j - s_{\sigma(i)}) + 2 \cdot \big(\min\{x_j, t_{\sigma(i)}\} - x_i\big) + |x_j - t_{\sigma(i)}| \\ & & +(x_i - t_{\sigma(j)}) \\ & = & bad_2 + 2 \cdot \big(\min\{x_j, t_{\sigma(i)}\} - x_i\big) \\ bad_1 & > & bad_2 \end{array}$$

Therefore, the labeling with the crossing between the po-leader and the op-leader is not minimal.

- If the op-leader is downward, and its vertical segment crosses the po-leader. (See Figure 4.4(b)). Similarly, we have:

$$s_{\sigma(j)} < s_{\sigma(i)} < y_j < y_i$$

$$x_i < t_{\sigma(j)} < x_j, t_{\sigma(i)}$$

With the same calculations, we obtain:

$$bad_1 = bad_2 + 2 \cdot \big(\min\{x_j, t_{\sigma(i)}\} - t_{\sigma(j)}\big) > bad_2$$

- If the op-leader is upward, and its horizontal segment crosses the po-leader. (See Figure 4.4(c)). We have then:

$$s_{\sigma(i)} < y_j < y_i < s_{\sigma(j)}$$

$$x_i < t_{\sigma(j)} < x_j, t_{\sigma(i)}$$

  After some calculations, we obtain:

$$bad_1 = bad_2 + 2 \cdot \big(\min\{x_j, t_{\sigma(i)}\} - x_i\big) + 2 \cdot (y_i - y_j) > bad_2$$

- Otherwise, the op-leader is upward, and its vertical segment crosses the po-leader. (See Figure 4.4(d)). We have then:

$$y_j < s_{\sigma(i)} < y_i < s_{\sigma(j)}$$

$$x_i < t_{\sigma(j)} < x_j, t_{\sigma(i)}$$

  After some calculations, we obtain:

$$bad_1 = bad_2 + 2 \cdot \big(\min\{x_j, t_{\sigma(i)}\} - t_{\sigma(j)}\big) + 2 \cdot (y_i - y_j) > bad_2$$

We conclude that a minimal labeling contains no crossing between an op-leader and a po-leader.

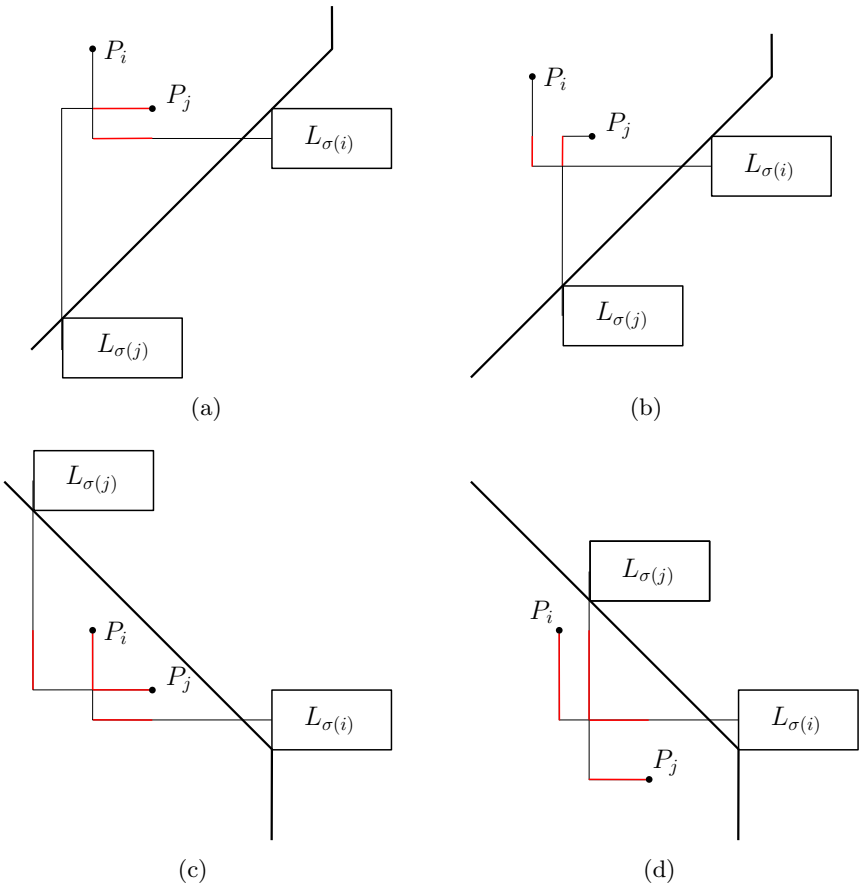
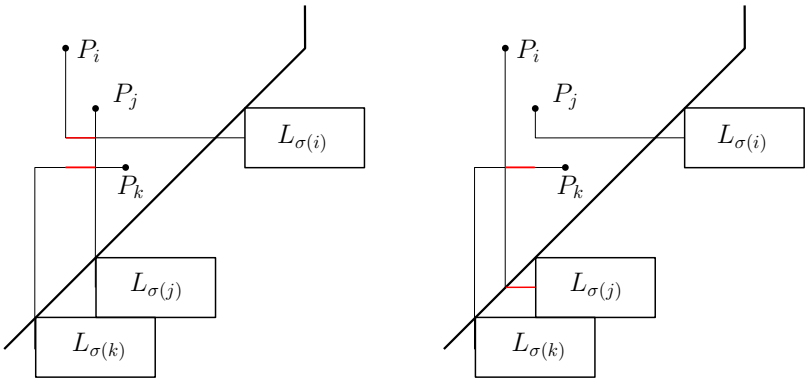


Figure 4.4: Different possibilities for crossings. The part of the leader highlighted in red represents the reduction of leader length by switching the two labels.

We now prove that a vertical leader cannot have a crossing with an op-leader as well as with a crossing with a po-leader in a minimal labeling, as illustrated in Figure 4.5. Suppose that

a vertical leader $\ell(P_j, L_{\sigma(j)})$ crosses a po-leader $\ell(P_i, L_{\sigma(i)})$ and an op-leader $\ell(P_k, L_{\sigma(k)})$. Then, switching the labels $L_{\sigma(i)}$ and $L_{\sigma(j)}$ does not change the total leader length but creates a crossing between an op-leader $\ell(P_k, L_{\sigma(k)})$ and a po-leader $\ell(P_i, L_{\sigma(j)})$. Since then we could improve the labeling, as proved above, the initial labeling is not optimal.



(a) Initial SAB-labeling

(b) After switching labels $L_{\sigma(i)}$ and $L_{\sigma(j)}$

Figure 4.5: Vertical leader crossing a po-leader and an op-leader. The parts of the leader highlighted in red represent the reduction of leader length by switching the labels.

Finally, we note that no SAB-labeling contains any leader-label crossing, because those crossings were caused by vertical leaders connected to shifted labels. However, we use op-leader here instead, which are drawn entirely inside the map. □

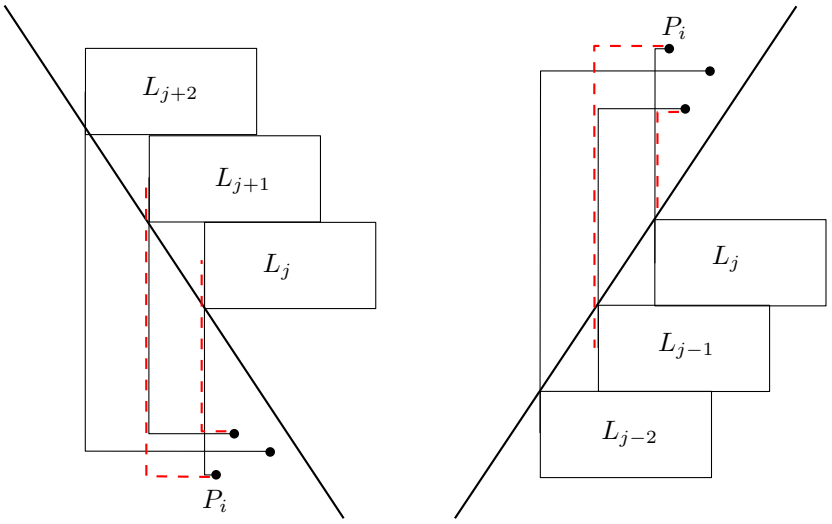
We finally present an Algorithm 16 removing crossings from a minimal labeling without increasing its leader length. Figure 4.6 shows what happens in this this algorithm.

Algorithm 16: Algorithm removing the po-leader crossings

Data: Order σ of the labels, coordinates of the points $P_i : (x_i, y_i)$ and $L_j : (t_j, s_j)$
Result: Gives the y-coordinate of the label s'_0 corresponding to a possibly minimal cluster SAB-labeling

```
// removing crossing involving downward op-leaders
for j from n − 1 to 0 do
    i = σ^{-1}(j);
    if ℓ(P_i, L_j) is a downward op-leader and crosses another leader then
        L_k := label connected to an op-leader with the bottommost intersection with
        ℓ(P_i, L_j);
        Switch L_j and L_k.;
// removing crossing involving upward op-leaders
for j from 0 to n − 1 do
    i = σ^{-1}(j);
    if ℓ(P_i, L_j) is an upward op-leader and crosses another leader then
        L_k := label connected to an op leader with the topmost intersection with
        ℓ(P_i, L_j);
        Switch L_j and L_k.;
```

Theorem 13. *Given a minimal SAB-labeling of n points. Algorithm 2 removes the crossings between po-leaders in $O(n^2)$ time and Algorithm 16 removes the crossings between op-leaders in $O(n^2)$ time.*



(a) between upward leaders (b) between downward leaders

Figure 4.6: Illustration of the crossing removal for op-leaders. The dashed strokes represent the new leaders after switching the labels.

This theorem can be proved with the same argumentation as Theorems 11 and 7

4.1.5 Conclusion

In the previous sections, we studied algorithms to compute weakly-aligned boundary labelings, with minimal leader length and the constraint that labels must be on the right of the convex border. In this model, it is possible that some labels do not touch the border, and are then shifted. If too many labels in a same cluster are shifted, it may be more difficult to follow the leaders with the eyes.

In this section, we studied an alternative model with an additional constraint to require labels to touch the border. The new op-leader type must then be allowed to connect a point to a label to the left of it. This new labeling type is called *SAB-labeling*, or *strictly-boundary aligned labeling*

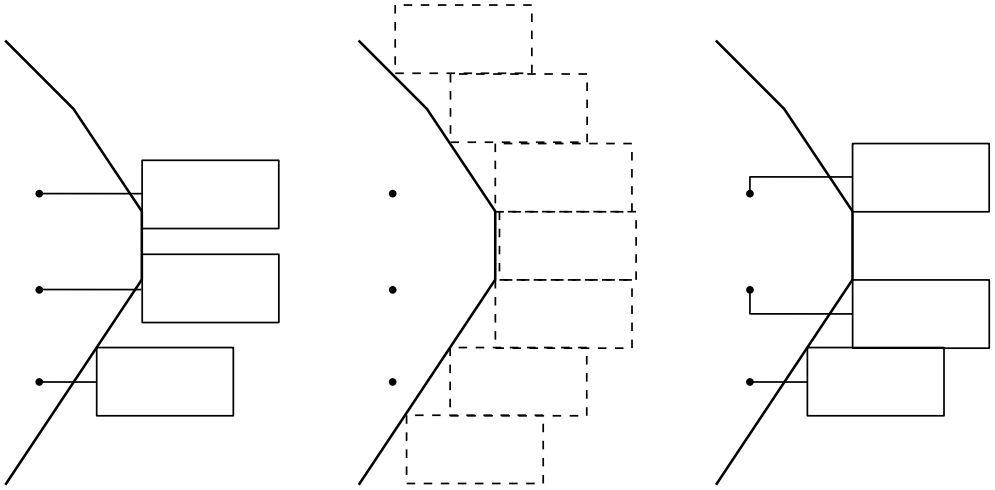
A *SAB-labeling* has the same properties as a WAB-labelings except for two differences. First, the shifted labels present in WAB-labelings are here replaced with label connected by op-leaders. As a consequence, the leader-label crossings are replaced by crossings between two op-leaders. These two types of crossing are easy to remove. For a point and a border, the leader length function is not exactly the same in both types of labelings. In SAB-labelings, the length of the leader increases more rapidly when the label is connected to an op-leader than it does in WAB-labelings when the label gets shifted. Therefore, the minimal labeling of a cluster may not have the same position for the two labelings.

Finally, the algorithm used to compute an optimal SAB-labeling is the same as in previous section, after changing the leader length functions and replacing Algorithm 11 with Algorithm 16 in order to remove the op-leader crossings instead of the leader-labels crossings.

4.2 Discrete Labeling using ILP.

In this section, we consider another method to compute a minimal labeling either using po-leader and allowing shifted labels or using po- and op-leaders and forcing labels to be on the border. we suppose then that we compute either a weakly-aligned boundary labeling or a strictly-boundary aligned labeling.

In this section, we discretize the possible label positions, and try to avoid computing several matchings and cluster labelings. We denote by *slots* the possible positions for the labels. We assume that there are $m \geq n$ slots for the labels, and look for a specific matching to compute directly a minimum labeling. If the label slots do not intersect, a single minimum weighted matching provides a minimal labeling. However, the label positions may not be aesthetically satisfying. Figure 4.7 presents a minimal labeling where the positions of the m slots of the labels do not intersect. This labeling may irritate the user due to the fact that some leaders are not straight, but for different label position, the leader can be straight in an optimal labeling.



(a) Desired labeling (b) Slots for a discrete labeling (c) non-desired labeling

Figure 4.7: Labeling with fixed label slots. If the slots are chosen badly, the position of some labels is disturbing (Figure (a)). We desire to compute a labeling as presented in Figure (b). However, for such a labeling the slot of some labels may intersect.

Here, we allow label slots to intersect. The objective is to create a matching algorithm which takes into account the fact that the slots chosen by the matching must not intersect.

4.2.1 Integer Linear Program

In Section 3.1, we saw an integer linear program to compute a minimal labeling for a border consisting in a single edge. In section 3.2, we noted that the program from Section 3.1 can easily be adapted into an integer quadratic program. In this section, we formulate an integer linear program to compute a minimal labeling for a convex border, by modeling this problem as a matching with further constraints.

Let Eq be the equation of the border. Given a y-coordinate s_j of a label $L_j : (t_j, s_j)$, the x-coordinate of L_j is $t_j = \text{Eq}(s_j)$. We denote by $L_0, \ldots, L_{m-1}$ the slots of the labels sorted from the bottom to the top.

We recall the form of an ILP program:

$$\begin{aligned} &\text{minimize } c^T x \\ &\text{subject to } Ax \leq b \\ &\qquad\qquad x \geq 0 \end{aligned}$$

Variables.

Since the position of the m label is fixed, the variable to compute is only the matching $\sigma(\cdot,\cdot)$ associating a label to each point. Just as in Section 3.1, we create $n \cdot m$ variables $\sigma(i,j)$. For given indices i and j, $\sigma(i,j)$ is equal to 1 if the points P_i is connected to the label L_j, and 0 otherwise:

$$\text{Variables: } \sigma(i,j) \in \{0,1\}^{n^2} \text{ for } 0 \leq i,j \leq n-1 \tag{4.1}$$

Because the position of the labels is fixed, we do not need to add further variables as in Section 3.1 concerning the position of the labels and the leader lengths.

Optimization Function.

We create a matching where the cost of a connection between a point P_i and a label L_j corresponding to the variable $\sigma(i,j)$ is equal to the leader length $|\ell(P_i, L_j)|$.

$$\text{function to Minimize: bad} = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} |\ell(P_i, L_j)| \tag{4.2}$$

This function is characterized by the vector c, composed of $n \cdot m$ values corresponding to each leader length, sorted with respect to the order of the variables in x they correspond to.

Constraints.

We are looking for a minimal labeling, which means in particular that crossings are allowed. We will later use another algorithm to remove the crossings.

Since the position of the labels is already fixed, the only constraint is that no two labels may intersect. Denote by *Inter(j)* the set of indices of the labels intersecting L_j. Thus, the constraint of the labels in the matching is:

$$\text{Constraints: } \forall j, \sum_{i=1}^{n} \sum_{k \in \text{Inter}(j)} \sigma(i,k) \leq 1 \tag{4.3}$$

4.2.2 Choice of Slots

The integer linear program presented above provides a minimal labeling given a fixed position for the label slots. However, what we are looking for here is a minimal labeling with variable label positions. That means that no other label position can reduce the total leader length. In this part, we are detailing a choice of the slots such that the positions of the labels given by the integer linear program correspond to a minimal labeling.

For this purpose, we use Corollary 2 of Section 3.2 and Corollary 3 of Section 4.1 providing an interesting property regarding the positions of the labels in a particular minimal cluster labeling, and deduce some properties about the possible positions of the labels in a minimal labeling, and thus the placement of the slots in the integer linear program.

These two corollaries state for both types of labelings that we can always compute a minimal labeling such that every cluster contains a label satisfying one of the following statements:

- it is connected by a horizontal-straight leader,

- it is connected by an upward-straight leader,
- it is connected by a downward-straight leader,
- one of its corners touches one of the $k+1$ border nodes.

Therefore, there exists a minimal cluster labeling so that every label is placed either at a position described as above, or moved by a distance $i \cdot h$ upward (or downward) from it, where h represents the height of the labels, and $i \in \mathbb{Z}$ is an integer.

Furthermore, since every cluster contains at most n labels, we can reduce the possible values of the integer from i to $\{1-n, 2-n, \ldots, -1, 0, 1, \ldots, n-2, n-1\}$. Each point provides then $(k+4) \cdot (2n-1)$ positions for the slots. Indeed, suppose that a label L_j is placed at a y-coordinate s_j. Then, every label of the cluster must have a y-coordinate of the form $s_j + i \cdot h$, where h is the height of a label, and $1-n \leq i \leq n-1$.

Altogether, we create at most $(k+4) \cdot n \cdot (2n-1)$ label slots, and an integer linear program as created previously computes a minimal labeling for the given type (op-po-labeling or po-labeling).

4.2.3 Conclusion

In this section, we computed a minimal labeling using a minimum weighted matching in bipartite graph with specific constraints on the edges allowed in the matching. Given n points and a border with k edges, the bipartite graph contains $O(k \cdot n^2)$ vertices. However, because we allowed intersecting label slots whereas that the labels used in the minimum weighted matching must not intersect, we cannot use a matching algorithm to compute the minimal labeling this way. The solution is then computed using integer linear programming. It would be interesting to study if we can create a polynomial time algorithm to compute a minimum weight matching with constraints, i.e., by forbidding some edges to be simultaneously in the solution. This would allow us to handle intersecting label slots.

4.3 Rectangular Block Labeling

In the previous sections, we saw how to compute a crossing-free labeling with a minimal total leader length when the border is convex and composed of edges with big inclinations. However the fact that some labels may have to be shifted makes the computations more difficult. Indeed, the Y-order may only provide a sub-optimal crossing-free labeling, and in this case, we have to use complex algorithms to solve the labeling problem. Therefore, we look for another model of convex-boundary labeling which avoids this problem without making the labeling less aesthetically pleasing.

In this section, we assume that labels in the same cluster must have the same x-coordinate, (see Figure 4.8). Thus, switching two labels inside a cluster does not change the horizontal leader length. We call such a labeling *rectangular-block labeling*, or *block-labeling*. The interest of this model is to create labeling such that (1) the label positions depend on the shape of the convex border, and (2) there always exists a minimal labeling using the Y-order. Furthermore, we assume that the leader length functions correspond to a WAB-labeling.

However, another challenge appears in this model, as presented in Figure 4.9. If we keep the same badness function, a minimal labeling will be similar to a minimal labeling computed in Sections 3.1 and 3.2. and the only difference will be the distance between two labels. Indeed, when we search for an optimal labeling of two points whose optimal label positions intersect, regrouping them in a single cluster usually produces a bigger badness than separating them. When the two labels are grouped into a single cluster, one of the

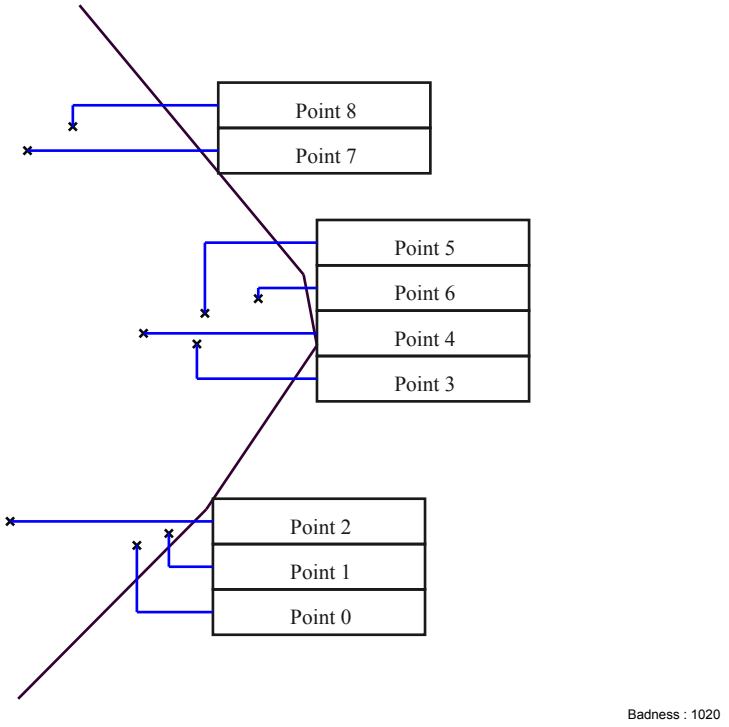
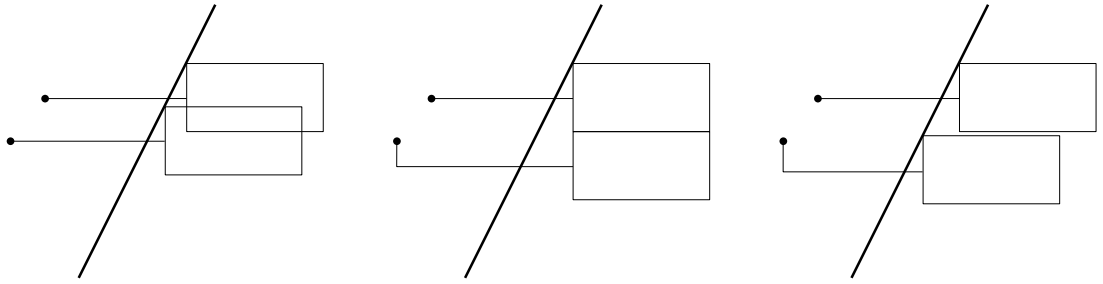


Figure 4.8: Example of a block-labeling. Every label inside the cluster must have the same x-coordinate.



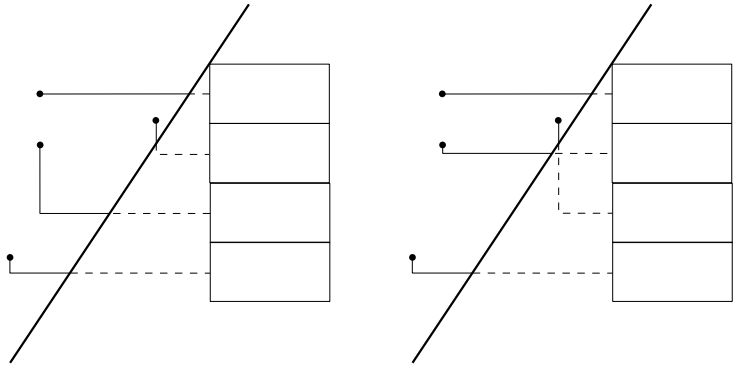
(a) Overlapping labels (b) Single cluster solution (c) Double clusters solution

Figure 4.9: The figures above shows the problem of applying the same badness function as before in the block-labeling. The figure (a) presents two overlapping labels. If we put them into the same cluster (Figure (b)), the total leader length will be greater than if we move the bottommost label downward to prevent the overlap. Therefore, it is not optimal to have two labels in the same cluster.

labels will be moved away to the right, and this horizontal distance is bigger than the vertical distance needed to separate the label in two different clusters.

A first solution would be to consider only leader length inside the map, i.e., the leader length left to the border. However, this creates a problem similar to the problem of a shifted label, as presented in Figure 4.10.

Our solution consists in keeping the same badness function as for the WAB-labeling, and adding a new constraint we call *local optimality*. This new constraint states that the position of a cluster of labels must be independent of the other clusters. Then, in Figure 4.9, the two labels have to be in the same cluster. Otherwise, one of the labels would not be at its optimal position, and removing the other cluster makes the remaining cluster labeling suboptimal.



(a) Y-order (b) optimal order

Figure 4.10: Illustration of the problem similar to the shifted labels, when the badness only takes into account the leader length inside the map. In the figures, a leader that has no horizontal segment inside the map will have the same length if we connect it to any label below the current one. Therefore, the labeling from Figure (*b*) has a badness smaller than the badness of the labeling from Figure (*a*).

From the new constraint of local optimality, we deduce the following lemma:

Lemma 18. *The clusters computed by the upward-downward algorithm induce a minimal labeling.*

Proof. Consider two clusters C_1 and C_2 which overlap when placed at their respective optimal position. Then, in a minimal labeling, at least of of those clusters is not at its optimal position. Without loss of generality, let C_1 be not at its optimal position. Suppose now that they have not been regrouped into a single cluster C. Removing every labels of the cluster C_2 results in having the cluster C_1 back at its optimal position, which refuses the constraint of local optimality. □

A block-labeling is defined so that every label in the same cluster has the same x-coordinate. Therefore, instead of shifted labels, we here can have *shifted clusters*. Indeed, given a x-coordinate t for the labels in the cluster, a point P_i may have a x-coordinate x_i greater than t. Then, the cluster must be shifted to the right so that each label has a x-coordinate x_i. The following lemma gives a sufficient condition to prevent shifted clusters.

Lemma 19. *Consider a cluster of labels and the points connected to a label in this cluster. If the topmost point is below the topmost label and the bottommost point is above the bottommost label, then the cluster is not shifted.*

Proof. Consider a shifted cluster. The x-coordinate of its labels is x_i, where $P_i = (x_i, y_i)$ is the rightmost point connected to a label of the cluster. This point P_i is then either below the bottommost label or above the topmost label of the cluster. Otherwise, it would be connected to a label with a non-vertical leader, which contradicts the fact that the cluster is shifted. □

The conditions for this lemma are in particular realized when the clusters are created with the upward-downward algorithm. Indeed, if a point P_i is above the topmost label of a cluster, then switching the label it is connected to with the topmost label either reduces the total leader length if the topmost label is connected by an upward leader, or does not

change the total leader length otherwise. Therefore, we can assume that the point P_i is connected to the topmost label of the cluster. Then, we reduce the total leader length by putting the topmost label out of the cluster to the optimal position for the point P_i. The upward-downward algorithm would not have put the point P_i into this cluster.

The following lemma generalizes a result of the WAB-labeling for the case of the block-labelings.

Lemma 20. *The length function of a leader depending on the y-coordinate of the label is it connected to is piecewise linear and does not contain any local non-global minimum. The total leader length function is also piecewise linear, and the endpoints of its segments correspond to horizontal-, upward- or downward-straight leaders.*

Proof. Since the difference between a WAB-labeling and a block-labeling can only be found in the position of labels when they are in the same cluster, the function giving the length of a leader from the y-coordinate of its label is the same for both labelings. Lemma 5 states that these functions are piecewise linear and contain no local non-global minimum, the leader length function for a cluster also has these properties.

Moreover, let $\{L_0, \ldots, L_{n-1}\}$ be a cluster of n labels. In a block-labeling the y-coordinate of each label L_i is the same as in the WAB-labeling, and its x-coordinate is shifted by a distance of d_i to the right of the x-coordinate in the WAB-labeling. The value of this distance d_i is either $(n-1-i)\cdot h/m$ or $i \cdot h/m$, depending on whether the edge the cluster touches has positive or negative slope. Let $F(s_0)$ be total leader length for the WAB-labeling when the bottommost label has the y-coordinate s_0. The badness for the block-labeling is then $F_b(s_0) = F(s_0) + \left(\sum_{0 \leq i < n} i\right) \cdot h/m$. Since $F(s_0)$ is piecewise linear and the endpoints of its segments correspond to horizontal-, upward-, downward-straight leaders and the touching of labels corners with border nodes. $F_b(s_0)$ has the same property. □

In a minimal cluster labeling, the conditions given by Lemma 19 are always true. For example, if the topmost label of a cluster was connected by a downward leader, then moving its position upward would decrease the badness of the labeling. Because of Lemma 18, we will use the upward-downward algorithm to compute the clusters, and thus the clusters in a minimal labeling cannot be shifted. From this fact, we deduce the following corollary.

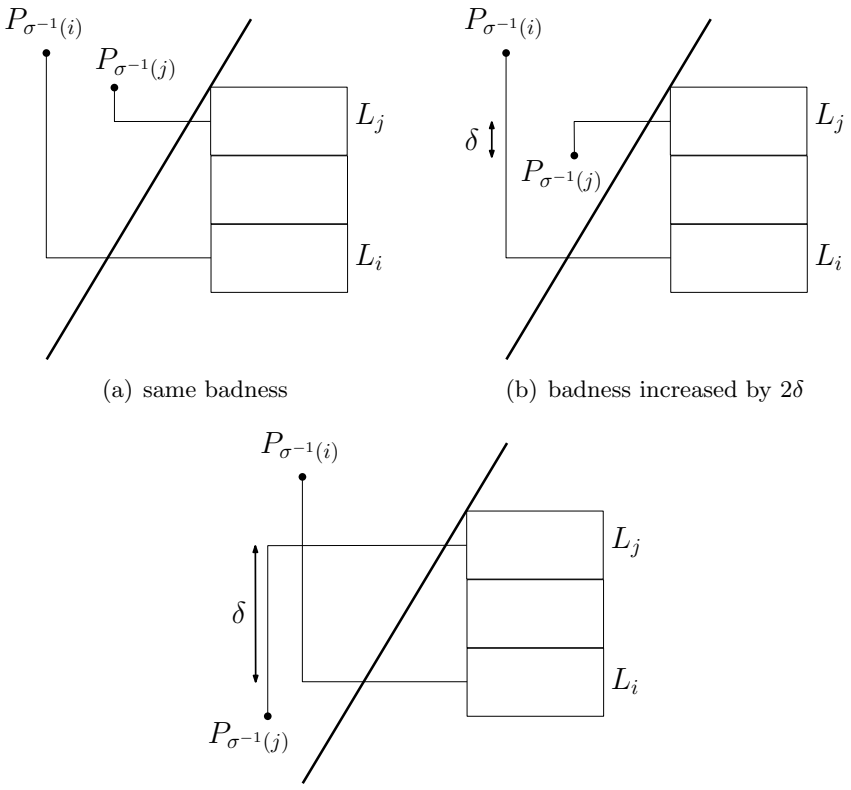
Corollary 4. *There exists a minimal cluster labeling containing a horizontal-straight leader, or so that a corner of a label overlays a border node. Moreover no minimal labeling contains downward-straight leaders or upward-straight leaders.*

Proof. Consider a cluster of labels. We know from Lemma 20 that the function giving the total leader length from the y-coordinate of the cluster is piecewise linear, and that the endpoints of the segments composing the function correspond to upward-straight, downward-straight and horizontal-straight leaders. However, when the conditions of Lemma 19 are realized, no point can be above or below the cluster. In particular, no leader can be vertical. We conclude that there must be a minimal cluster labeling containing a horizontal-straight leader. □

Furthermore, we do not have the problem created shifted labels that we encountered in a WAB-labeling. This leads the the following property of a block-labeling.

Lemma 21. *There exists a minimal block-labeling induced by the Y-order.*

Proof. Consider a minimal cluster block-labeling, using an order of labels σ. We suppose that the cluster has been computed with an upward-downward algorithm, and therefore we know from Lemma 19 that the cluster is on the border. Thus, for a fixed position of the cluster, the horizontal leader length is independent of the order of the labels σ. Consider two labels L_i and L_j with $i < j$ which do not respect the Y-order. They are connected to points $P_{\sigma^{-1}(i)}$ and $P_{\sigma^{-1}(j)}$ so that $\sigma^{-1}(i) > \sigma^{-1}(j)$. Switching labels L_i and L_j does not increase the total leader length, as illustrated in Figure 4.11.



(a) same badness

(b) badness increased by 2δ

(c) badness increased by 2δ

Figure 4.11: Switching two labels which do not respect the Y-order do not reduce the total leader length of a block-labeling. The three figures below illustrate the possible cases, depending on the position of $P_{\sigma^{-1}(j)}$ regarding L_i and L_j, and the change of badness when the labels L_i and L_j are being switched.

Therefore, iteratively switching labels which not respect the Y-order from the order σ to the Y-order will not increase the badness of the labeling. This means that, given a fixed cluster position, the Y-order minimizes the badness. Finally there exists a minimal block labeling using the Y-order. □

In a minimal labeling, leader-crossings can only happen between leaders connected to labels from a same cluster. moreover, we saw in sections 3 and 4 that leader-label crossings always involve downward-straight or upward-straight leaders. Nevertheless, Corollary 4 states that this cannot happen in block-labelings. Thus, given a minimal labeling of a single cluster, the crossing-removal problem is the same as for a *po*-right labeling for a vertical border. This leads to the following theorem:

Theorem 14. *Given a minimal labeling, Algorithm 2 removes the crossing from a minimal labeling in $O(n^2)$ time.*

Thus, all we have to do now is to find out how to compute a minimal cluster labeling.

4.3.1 Border With a Single Edge

First we suppose that the border consists of a single edge with a big, positive slope. The equation of the x-coordinate of a label $L_i : (t_i, s_i)$ is then $t_i = \frac{s_i}{m} + a$, with $m > 1$.

The following lemma gives a results from Algorithm 4 when searching for an optimal position of a single cluster for the Y-order, given the border equation and the coordinates of n points whose labels have to be placed in the same cluster.

Lemma 22. *The cluster position provided by Algorithm 4 for the Y-order induces a minimal block-labeling.*

Proof. This proof repeats many points already seen in *Lemma* 6. For a positive slope, the cluster is positioned so that the top-left corner of the topmost label is in the border. Since the topmost label can not be connected by a downward leader, every point is below the topmost label. Therefore, all the points are placed left to the labels and there can not be any shifted cluster.

From Corollary 4 and the fact that for each index i it holds that $x_i < t_i$. We know that there is a minimal cluster labeling with a horizontal straight leader.

Let's position the cluster characterized by a value s_0 for the bottommost label such that $y_{n-1} \leq s_{n-1}$ and there is a horizontal straight leader. First, move the cluster by a distance of ε upward so that no downward leader becomes upward. Then the total horizontal leader length is increased by $\frac{\varepsilon}{m}$ and the total vertical leader length is increased by $\varepsilon\big(n_u + n_s - n_d\big)$, where n_u, n_s and n_d are the number of upward, horizontal-straight and downward leaders respectively. The upward gain is then: $G_u(s_0, \varepsilon) = \varepsilon\big(\frac{n}{m} + n_u + n_s - n_d\big) = \varepsilon\big(\frac{n}{m} + 2 \cdot n_u + 2 \cdot n_s - n\big)$. Likewise, we compute the downward gain $G_u(s_0, \varepsilon) = \varepsilon\big(-\frac{n}{m} - 2 \cdot n_u + n\big)$.

For a single edge border, the length function of a leader depending on the y-coordinate of its label is convex. Therefore, any local minimum is global, and the minimum of the convex total leader length function is attained when both $G_u(s_0, \varepsilon)$ and $G_d(s_0, \varepsilon)$ are positive for sufficiently small values of ε:

- $G_u(s_0, \varepsilon) \geq 0 \Rightarrow \frac{n}{m} + 2 \cdot n_u + 2 \cdot n_s - n \geq 0 \Rightarrow n_u \geq \frac{n}{2}\big(1 - \frac{1}{m}\big) - n_s$
- $G_d(s_0, \varepsilon) \geq 0 \Rightarrow -\frac{n}{m} - 2 \cdot n_u + n \geq 0 \Rightarrow n_u \leq \frac{n}{2}\big(1 - \frac{1}{m}\big)$

An optimal position corresponds then to $n_u \in [\frac{n}{2}\big(1 - \frac{1}{m}\big) - n_s, \frac{n}{2}\big(1 - \frac{1}{m}\big)]$. Let ν be the unique number in $[\frac{n}{2}\big(1 - \frac{1}{m}\big), \frac{n}{2}\big(1 - \frac{1}{m}\big) + 1]$ and k be the index of the point which has the ν smallest value of $y_i - i \cdot h$. If we position the cluster so that the leader incident to P_k is horizontal straight, then there will be $k - 1$ leaders upward or straight and $n - k$ leaders downward or straight. The number of upward leaders will be at most $k - 1$ and at least $k - n_s$. Since this interval includes all integer values in $[\frac{n}{2}\big(1 - \frac{1}{m}\big) - n_s, \frac{n}{2}\big(1 - \frac{1}{m}\big)]$, This cluster position is optimal. This cluster position is returned by Algorithm 4, which concludes the proof. □

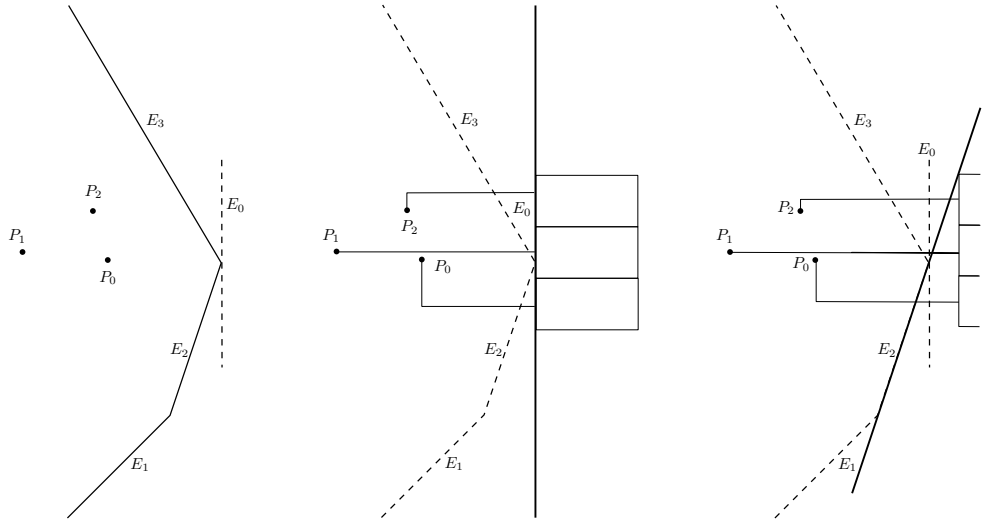
Corollary 5. *Given a cluster of labels and for a border consisting in a single edge, one of the minimal cluster labelings contains a horizontal-straight leader.*

4.3.2 Border With a Several Edges with Big Slopes

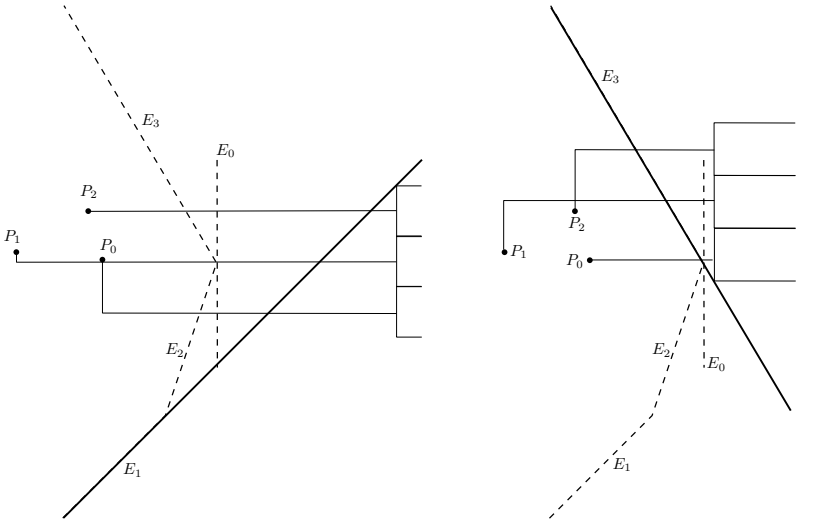
We suppose now that the border is convex and composed of k edges with big slopes $|m_k| > 1$. We denote by E_j the equation of the x-coordinate t_i of a label L_i from its y-coordinate s_i when the label is placed along the edge with index $j \in \{0, \ldots, k-1\}$: $t_i = \frac{s_i}{m_j} + a_j$. We denote by E the equation for the border.

When a label is on an edge with positive slope, its top-left corner is on the border. Similarly, when a label is on an edge with positive slope, its bottom-left corner is on the border. However, when the border contains edges with positive and edges with negative slopes, there are positions between them where neither the top-left corner nor the bottom-left corner of the label is on the border. When this occurs, the x-coordinate of the label is equal to the x-coordinate of the rightmost note.

The following lemma gives a simple method to compute a minimal labeling for a border with k edges $E_1, \ldots, E_k$ and a virtual edge E_0 (see Figure 4.12).



(a) Border (b) Labeling with E_0 (c) Labeling with E_2



(d) Labeling with E_1 (e) Labeling with E_3

Figure 4.12: A minimal labeling of the cluster for a convex boundary is the best of the labeling computed by taking each edge of the border independently.

Lemma 23. *One of the minimal cluster labelings corresponds to the best positions among the $k+1$ optimal position generated by considering that the border has the equation E_i for $i \in \{0, \ldots, k\}$.*

Proof. Consider a minimal cluster labeling. In this position, the cluster is on one of the k borders. Otherwise, we can reduce its badness by moving the cluster on the left. Let E_i, $i \in \{0, \ldots, k\}$ be such a border.

Consider a minimal cluster labeling supposing that the border equation is E_i. By definition, the badness of this new cluster position is smaller or equal to the badness of the minimal cluster labeling for the border E. If no corner of the cluster in this position was on the border E, then there would be another cluster position on the border with a strictly smaller badness. Moreover, this other cluster position will have a badness strictly smaller than the badness of the minimal cluster labeling, which is a contradiction. Therefore, the minimal cluster labeling for the border E_i is the minimal cluster labeling for the border E. We conclude that the minimal cluster labeling for the border E is one of the $k+1$ optimal positions supposing respectively that the border is E_i. □

Let h be the height of the labels, m the slope of the border edge and Y' be a sorted list of elements $(y_i - i \cdot h, i)$. Let $\{E_0, \ldots, E_k\}$ be the equations of the edges composing the border. The following algorithm computes an optimal position for the cluster characterized by the y-coordinate of the bottommost label of the cluster:

Algorithm 17: Position of a Cluster of n labels, for a Block-labeling

Data: Points $\{P_0, \ldots, P_{n-1}\}$, $\{E_0, \ldots, E_k\}$, optional: Y',
Result: Position s_0 of a minimal cluster labeling
s_0 := array of $k+1$ elements;
badness:= array of $k+1$ elements;
foreach $i \in \{0 \ldots k\}$ **do**
  ν_i = unique integer value in $[\frac{n}{2}(1 - \frac{1}{|m|}), \frac{n}{2}(1 - \frac{1}{|m|}) + 1[$;
  if $m_i > 1$ **then**
    $(y_k - k \cdot h, k) = \nu_i^{th}$ first element in Y';
  else
    $(y_k - k \cdot h, k) = \nu_i^{th}$ last element in Y';
  badness$[i]$:= badness of the cluster labeling at the position $y_k - k \cdot h$;
  $s_0[i] = y_k - k \cdot h$,;
return $s_0[i]$ that minimizes badness$[i]$;

Corollary 6. *One of the minimal cluster labelings for a cluster of labels and given a border equation for $k \geq 1$ edges contains a horizontal-straight leader.*

Theorem 15. *Algorithm 17 computes in $O(n \log n + n \cdot k)$ time a minimal cluster labeling in general case, and in $O(k \cdot n)$ time if the list Y' is provided.*

Proof. Lemma 23 shows the correctness of the algorithm. It remains to show the time complexity.

Computing and sorting the list $Y' = \{(y_i - i \cdot h, i) \mid i = 0 \ldots n-1\}$ takes $O(n \log n)$ time. Algorithm 17 computes k cluster labelings. For each of them, the cluster position in computed in constant time and the badness in $O(n)$ time. Finding the edge corresponding to a minimal cluster labeling is done by a minimal badness search in $O(k)$ time. The total time complexity to compute the minimal cluster labeling corresponding to one of these cluster position is then $O(n \cdot k)$, supposing that the list Y' is already computed. □

We deduce the following algorithm to compute an optimal labeling:

Algorithm 18: Optimal Block-Labeling for Single-Edge Border with slope $m > 1$

Data: Points $P_0, \ldots, P_{n-1}$
Result: optimal labeling.
Compute a minimal cluster labeling with Algorithm 1, given in input Algorithm 17 to provide a minimal cluster labelings;
Call Algorithm 2 to remove leader crossings;
return the labeling;

Theorem 16. *Algorithm 18 computes in $O(k \cdot n^2)$ an optimal labeling.*

Proof. The correctness of the algorithm follows from Lemmas 18 and 23. It remains to show the time complexity.

Given two overlapping clusters of labels C_1 and C_2 described by the lists Y_1' and Y_2', the Upward-downward algorithm enables to compute the list Y' in linear $O(n^2)$ time, and Algorithm 17 access in constant time a single value in Y', in oder to compute in $O(n \cdot k)$ time a cluster labeling. The Upward-downward algorithm call (n) time Algorithm 17, thus computing a minimal labeling has then a total time complexity of $O(k \cdot n^2)$.

Finally, with Algorithm 2, the crossings can be removed in $O(n^2)$ time. □

Algorithm 18 computes thus an optimal labeling in quadratic time in the number of points, which is as fast as the results provided by Theorem 3, with a factor linear in the number of edges.

4.3.3 Conclusion

The two previous parts presented results and algorithms to compute an optimal labeling with minimal leader length, given a set of n points, and a convex border E delimiting the leftmost position for the labels. These results generalize the work done by Nöllenburg et al. [NPS10] on one-sided labeling for vertical boundary.

However, this model presents several algorithmic problems which strongly increases the time complexity. For a general convex border, the complexity is in $O\big(n^5 \cdot (\log n)^3 \cdot 2^k\big)$, in comparison to $O(n^2)$ when the border is vertical. In aligned boundary labelings, the factor 2^k can be removed by adding a further constraint of local optimality and using the Upward-downward algorithm.

This complexity can be explained by two new problems. First of all, some labels can be shifted, and in this case it takes a lot of time to determine how the labels have to be ordered along the y-coordinates. Moreover, the presence of several edges makes the upward-downward algorithm less efficient to compute the repartition of labels in clusters.

In this section, We considered the *block-labeling*, which is a new model of the convex boundary labeling. The *block-constraint* forces the labels inside the same cluster to have the same x-coordinate. Hereby, the problem is more similar to the case of a vertical border, and no label can be shifted anymore. The second constraint is a *local-optimality* constraint, and forbid the cluster positions to depend on each others. This strong condition enables not only to keep the simple badness function as before, but also enables the Upward-downward algorithm used in Sections 2.4 and 3.1 to compute in this new model an optimal clustering.

We conclude this section by mentioning that, due to the strong constraint of local optimality for the labels, the algorithms for block labeling can be extended for two-sided labeling

and for convex border including small slopes. Nonetheless the Y-order may not induce a minimal cluster labeling when we consider edges with small slopes. Another possible extension for block labeling involving the idea of multi-sided labeling would be to consider non-convex borders.

5. Evaluation

In the following, we present some experimental results of our labeling algorithms. In the past sections, we studied the strictly- and weakly- boundary aligned labelings, and the rectangular-block labeling. This chapter presents experimental results of the different labeling programs.

The first section describes the algorithms that will be studied in the later sections. In section 5.2, the difference of visual quality of the labeling algorithms will be studied. Even though we primarily focus on the esthetic of a labeling, it is also important that the labeling algorithm are computed in a reasonable run time. In the last section, we compare the run times of each algorithm for different border shapes and different number of points. We will in particular confront the labelings for vertical boundary presented in section 2.4 to the complex algorithms to compute an optimal cluster boundary-aligned labeling and to the simplified rectangular-block labeling.

5.1 Description of the Labeling Programs

In this part, we present the different algorithms we will evaluate in this chapter.

ALGO_VB. This first algorithm computes an optimal labeling assuming that the right side of the boundary is vertical, and has been presented in Section 2.4. The time complexity of this algorithm is $O(n^2)$. However, we will only compare the quality of this labeling, and not its run time that we will assume equal to the run time of ALGO_Block described later.

ALGO_AB. This program computes an optimal aligned-boundary labeling. Here, we added to the problem a local optimality constraint of the solution so that the Upward-downward algorithm induces a minimal labeling. Moreover, we used the Hungarian algorithm [Kuh55] instead of the Algorithm of Vaidya [Vai89] to compute a minimum weighted perfect matching. Therefore, the worst complexity of ALGO_AB is $O(n^5 \cdot (n+k)\cdot)$ instead of $O(n^4 \cdot (n+k) \cdot (\log n)^3 \cdot 2^k)$. ALGO_AB regroups actually two different algorithms **ALGO_WAB** and algorithms **ALGO_SAB** computing respectively an optimal WAB-labeling and an optimal SAB-labeling. Both algorithms have exactly the same structure, call the similar algorithms and have exactly the same run time. We will on one side compare the quality of ALGO_WAB and ALGO_SAB, and on the other side compare the run time of the general algorithm *ALGO_AB* to the run time of others algorithms.

ALGO_SL. For this algorithm, we assume that the boundary in composed of several edges with big slopes, and computes a crossing-free WAB-labeling using the Upward-downward

algorithm and the Sweep-line algorithm Algorithm 10. A cluster labeling computed by Algorithm10 is known to be minimal if it contains no shifted labels. Therefore, a WAB-labeling computed by ALGO_SL is not necessarily minimal. The implemented algorithm takes $O(n^2 \log n)$ time to compute a labeling, but this complexity could have been improved to $O(n^2)$. The complexity of this algorithm is equal to the best case time-complexity of ALGO_AB. We want here to compare the run time of ALGO_AB to the run time of ALGO_SL representing its best-case complexity, and therefore find out how often shifted labels appear and how much they increase the run time.

ALGO_M. This algorithm is the same ALGO_SL, but uses the matching algorithm 9 instead of the sweep-line algorithm. As for ALGO_AB, we used here the Hungarian algorithm to compute a minimum weighted perfect matching. The complexity of *ALGO_M* is exactly the same as the worst-case time-complexity of *ALGO_AB*. while ALGO_SL shows how much the shifted labels increase the run time, we want to test with ALGO_M how efficient is the algorithm combining the sweep-line and the matching algorithms in comparison to the single use of the matching algorithm.

ALGO_Block. This algorithm computes with $O(k \cdot n^2)$ time-complexity an optimal block-labeling. Its run time should be close to the run time of ALGO_VB and it should be much faster than ALGO_AB. We will compare the run time and the visual quality between this algorithm and the two border-aligned algorithms.

In order to compute the labelings, a graphic user interface has been created to manually place points and border nodes on a map. The interface calls then the desired algorithm to compute the corresponding labeling. Moreover, the content and the size of the labels can be decided. Several Figures from in this thesis are screenshots of this interface. Figure 5.1 shows a screenshot of the interface.

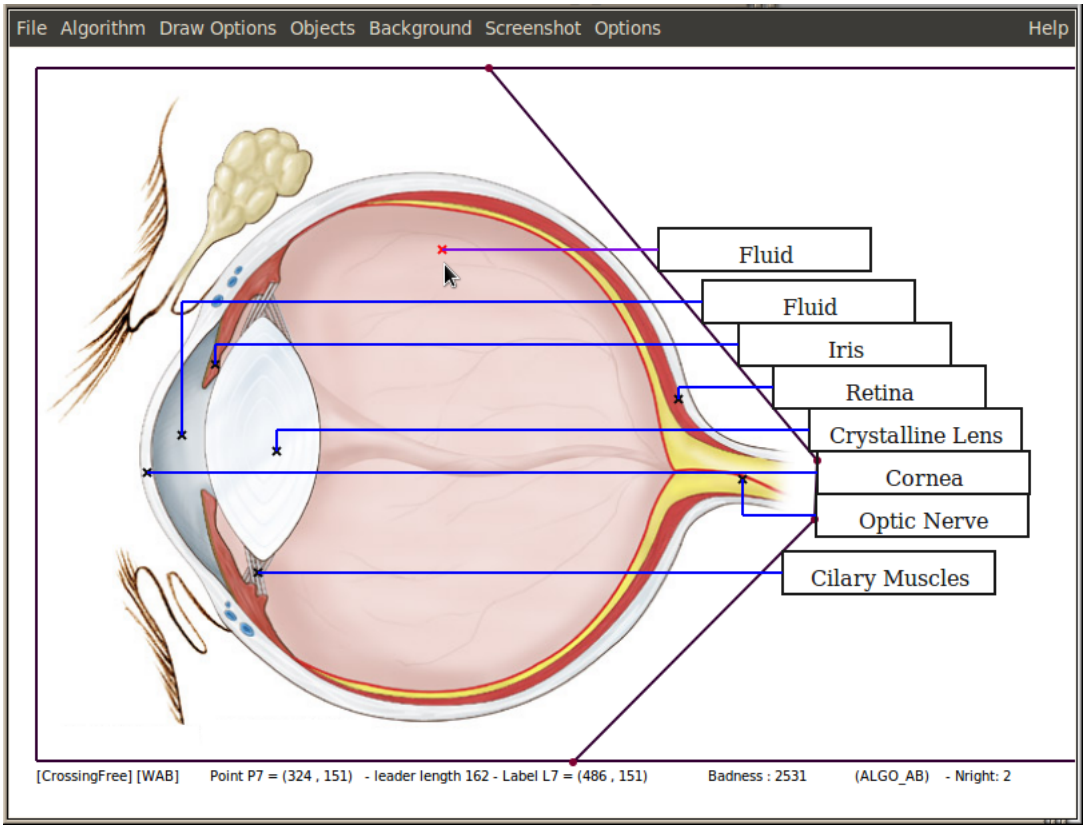


Figure 5.1: Graphic User Interface.

5.2 Quality of the Labelings

In this section, we compare the some labelings computed respectively by ALGO_VB, ALGO_WAB, ALGO_SAB, ALGO_Block. We show labelings with a small number of points to compare the visual quality of the different labeling algorithms. The following map are small enough to be easily read and the size of the labels is big enough to contain a text description. Moreover, the labels altogether do not occupy the whole height of the illustration.

We first present here a convex-boundary of the map of Germany (see Figure 5.2)

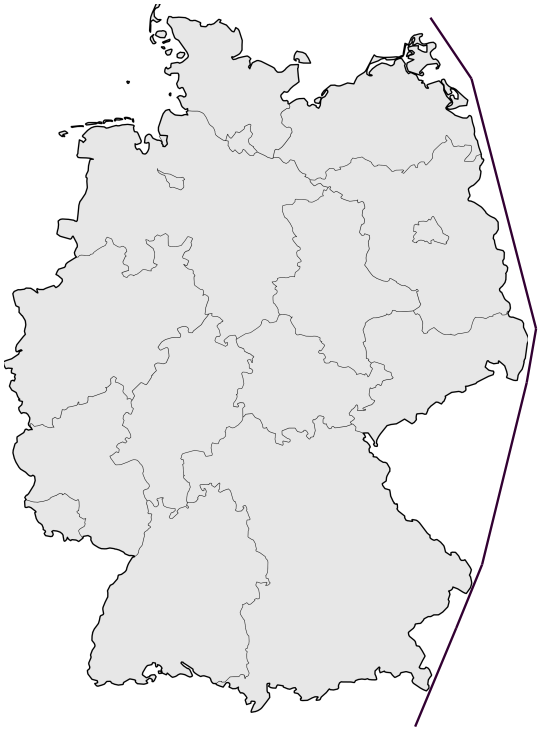


Figure 5.2: Map of Germany and the chosen convex boundary, without the points and the labels.

We compute a labeling to place a textual label describing each Land. The points representing are positioned arbitrary in the surface of the Länder. Note that the right border of Germany is close from being vertical, and that most Länder are far away to the left of the boundary. Therefore, we can already foresee that ALGO_AB will compute an optimal labeling without calling a matching algorithm. This gives the intuition that in most real maps, the time complexity of the boundary-aligned labelings corresponds to the best-case complexity.

Figures 5.3, 5.4 and 5.5 present the labelings computed with the different algorithms.

As expected, the labelings computed by ALGO_SL and ALGO_AB are exactly the same, and the position of the labels follows the shape of the border. Moreover, the block-labeling consists in several cluster close to the border of Germany, in opposite to the cluster placed by ALGO_VB that are floating further away from the border. Furthermore, the algorithms ALGO_SL, ALGO_AB and ALGO_Block computed labeling with the same five clusters, whereas the labeling computed by ALGO_VB only contains two clusters. Therefore, even for a border close to a vertical border, the slopes of the edges have an influence on the y-coordinates of the labels and thereby on the number of clusters.

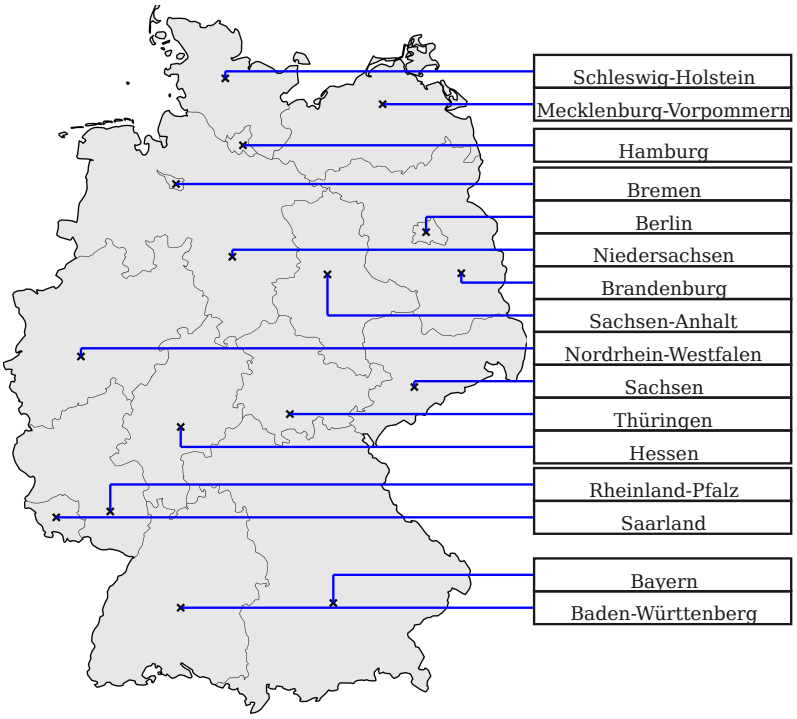


Figure 5.3: Boundary Labeling computed by ALGO_VB

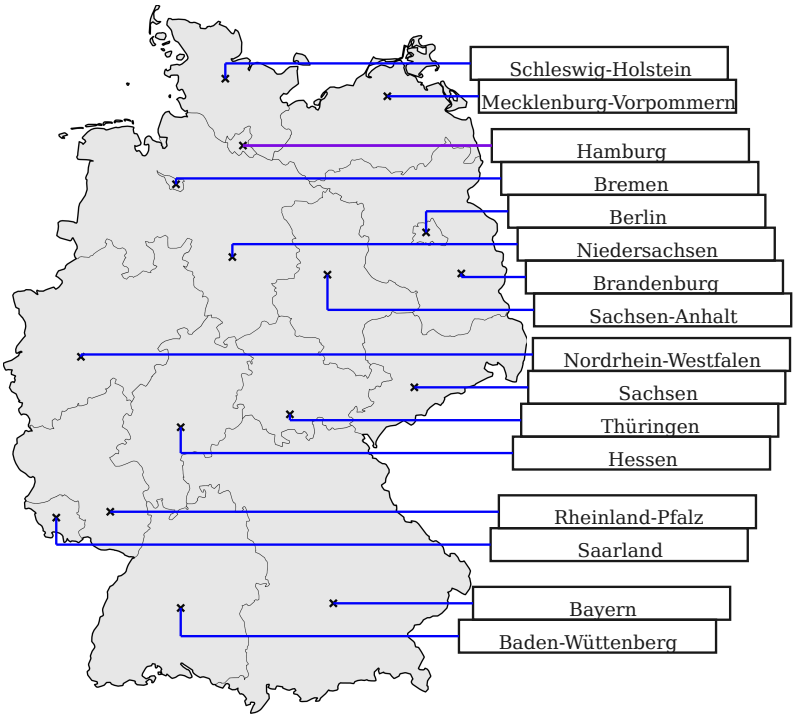


Figure 5.4: Boundary Labeling computed by ALGO_SL and ALGO_AB

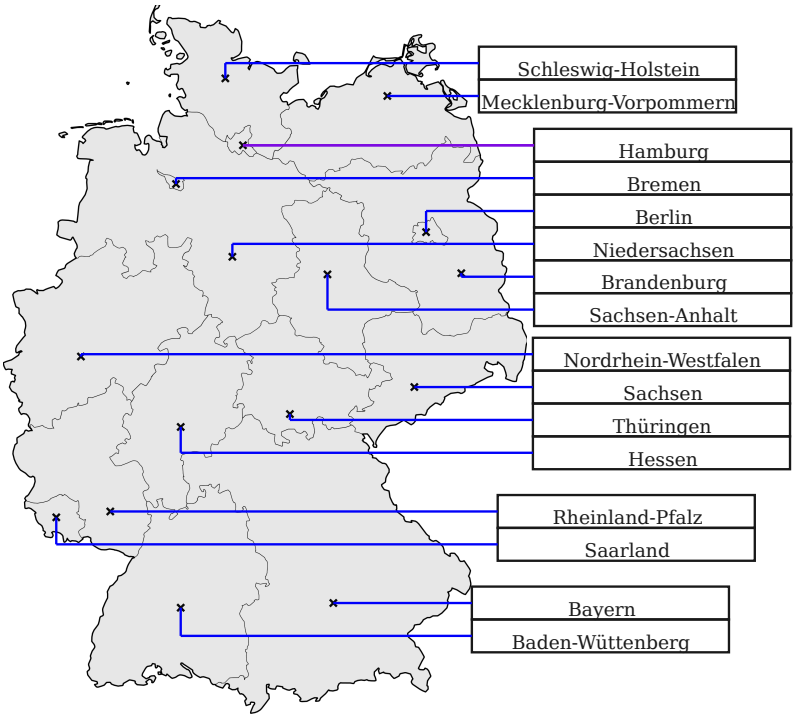


Figure 5.5: Boundary Labeling computed by ALGO_Block

We now present a map of Italy which has a different convex boundary. Each point represents a region of Italy and in each label is written the name of the corresponding region.

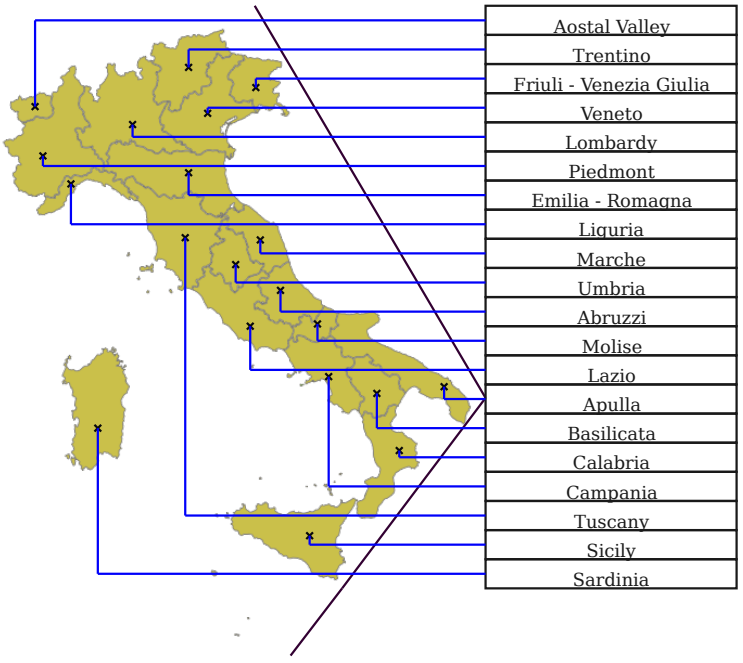


Figure 5.6: Boundary Labeling computed by ALGO_VB and ALGO_Block.

The right border of Italy consists of two edges with slopes close to 1. Therefore, there may be shifted labels or op-leaders in the boundary-aligned labelings.

Figures 5.4 and 5.5 present the labelings computed by the algorithms.

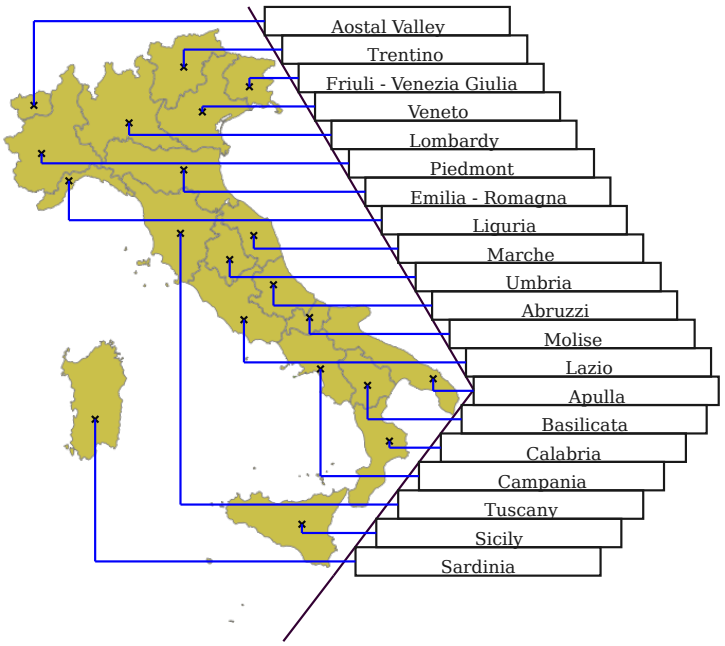


Figure 5.7: Boundary Labeling computed by ALGO_SL and ALGO_AB.

We first remark that, due to the high number of labels, the computed labelings are composed of a single cluster. As a consequence, the labelings computed by ALGO_VB for a vertical border and ALGO_Block are the same. Nevertheless, it is possible that these two algorithms compute different labelings, both consisting of a single cluster. This occurs when the cluster computed by ALGO_Block do not touch the rightmost border node.

A second observation is that ALGO_SL and ALGO_AB compute the same labeling. Actually, the Sweep-line algorithm in ALGO_SL computes a minimal labeling containing a shifted label that is removed with the crossings, see Figure 5.8.

The map of Italy proves that the condition stated by Lemma 13 is sufficient but not necessary for the computed cluster labeling to be minimal. We could develop further fast algorithms to compute possibly minimal cluster labelings. For example, we could prove that, if the cluster labeling computed by Algorithm 10 contains no shifted label after we remove the crossings, then it is minimal. We will see here that the approximation provided by ALGO_SL is usually good enough.

Since the WAB-labeling shown in Figure 5.8 contains a shifted label, we could compare it with the SAB-labeling version. We will not do that here, because there is only a single shifted label, and this label is not shifted far away from the boundary. Instead, we compare the WAB- labeling to the SAB-labeling in another map which do not represents real data. The labeling produced by ALGO_WAB and ALGO_SAB are shown in Figure 5.9.

We remark from the shape of leader connected to *Point* 5, that the two labelings do not have exactly the same cluster position. Since the leader length function increase more rapidly in SAB-labeling when the leader has type op, the cluster position computed by ALGO_SAB is slightly above the cluster position computed by ALGO_WAB. The preferred

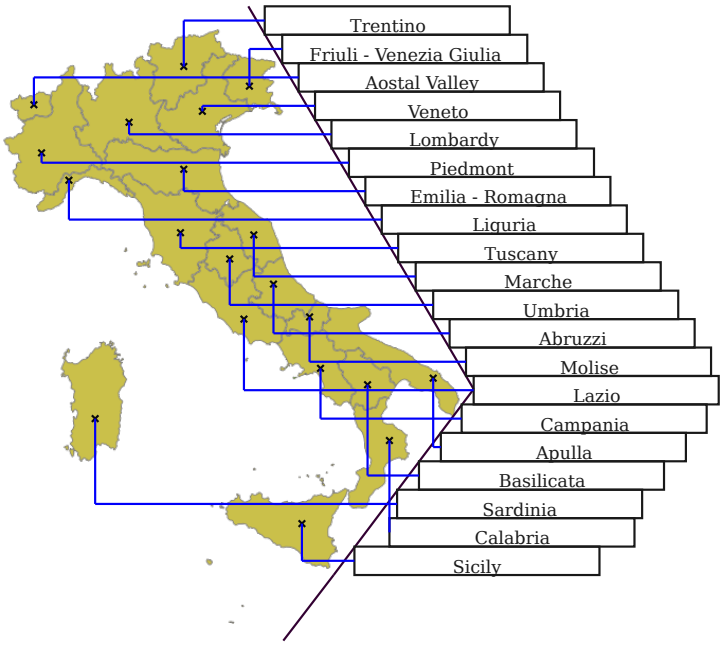
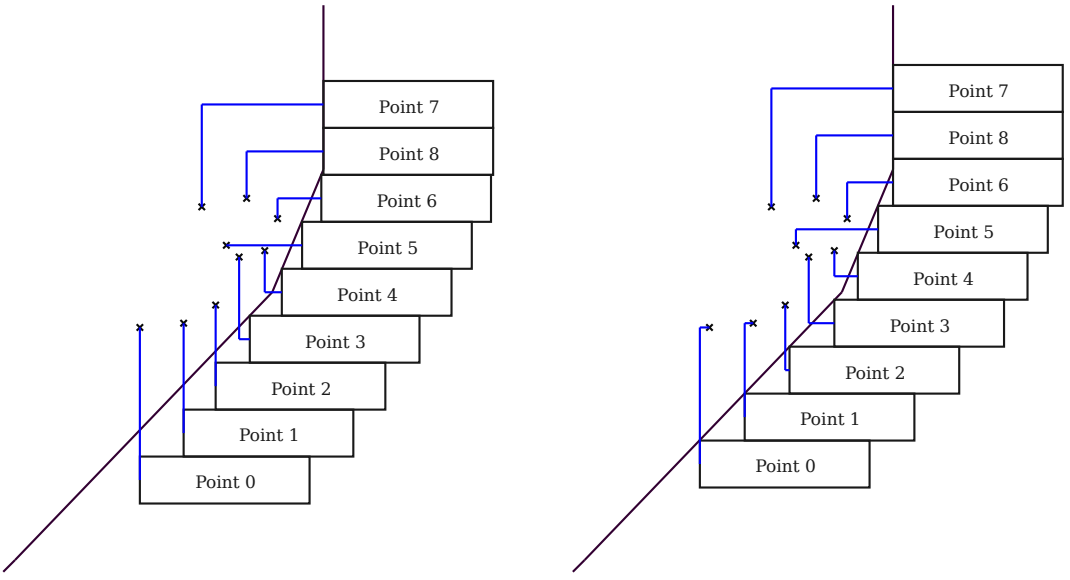


Figure 5.8: Minimal Labeling computed by ALGO_SL. The second bottommost label, named *Calabria*, is shifted. At the same time, it is connected by a crossing leader. After removing the crossings, the final labeling contains no shifted label (see Figure 5.7).



(a) Computed by ALGO_WAB (b) Computed by ALGO_SAB

Figure 5.9: Boundary Labeling computed by the aligned-boundary algorithm and containing shifted labels or op-leader.

of these two labelings will mainly depend on the user. We now present in Figure 5.10 the labeling computed by ALGO_Block and ALGO_VB for this map.

In this labeling, the three bottommost labels are considerably away from the border. Therefore, we could think about a variant of a block-labeling where each cluster can be

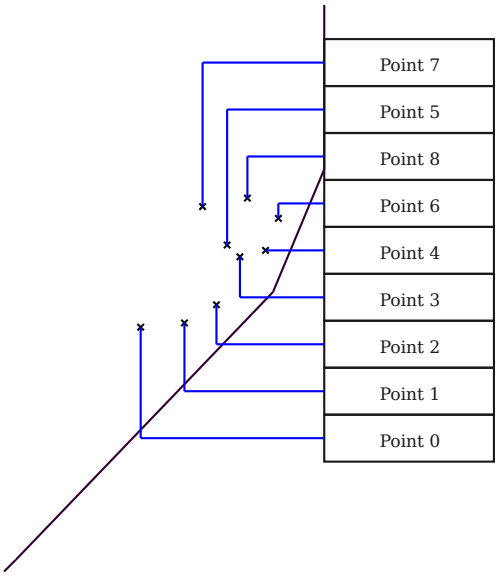


Figure 5.10: Boundary Labeling computed by ALGO_Block and ALGO_VB

broken into a few number of blocks, so that the labels in each block are vertically aligned.

Finally, we note that the WAB-labeling contains shifted labels because the points in the map are densely regrouped close to an edge of boundary whose slope is close to 1. Thus, the labels are grouped into a single cluster, and this cluster is placed further below the position computed in ALGO_VB. Consequently, shifted labels placed right below the points appear.

5.3 Run Time Analysis

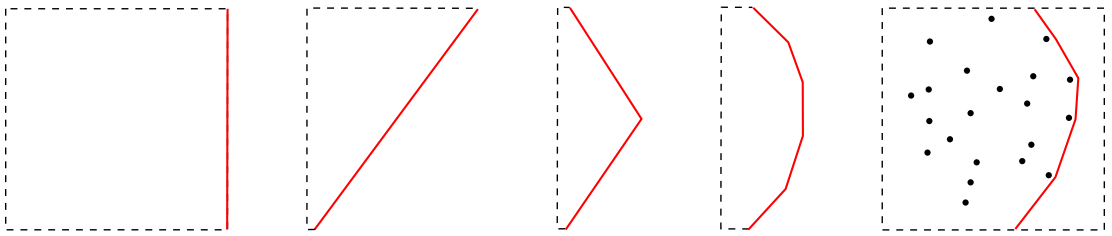
In the previous section, we visualized the different labeling models and compared some results. In this section, we consider the run time of four labeling algorithms:

- ALGO_SL computes a boundary-aligned labeling without using a matching algorithm. It corresponds to the best-case time complexity of Algorithm 14.
- ALGO_M computes a boundary-aligned labeling always calling $O(n^2)$ times a matching algorithm. This corresponds to the worst case time complexity of Algorithm 14.
- ALGO_AB is an implementation of Algorithm 14 which computes an optimal aligned-boundary labeling.
- ALGO_Block computes an optimal block labeling. This algorithm has the same structure as the algorithm ALGO_VB described by Nöllenburg et al [NPS10] to compute an optimal labeling for a vertical border. We will assume here that ALGO_VB and ALGO_Block have the same run time.

The main objective here is to compare the difference of run time between ALGO_SL, ALGO_M and ALGO_AB in order to find out what is the average time complexity of Algorithm 14, and to thus verify the intuition that there are no shifted label in most WAB-labeling. The second objective is to compare the run time of ALGO_Block to the algorithms for an aligned-boundary labeling. In this section, we first describe how we created the maps for our experiments, and then show some results.

Border Shape. In the previous sections, we saw that in maps where the points are not too much close to the boundary, ALGO_SL and ALGO_AB compute the same labeling, and thus the difference in run time should not be too big. In our experiments, we consider five different shapes of the boundary.

The first is a vertical border, denoted by VB, see Figure 5.11 (a). With this border, we expect that ALGO_SL and ALGO_AB have the same run time, and that ALGO_M is much slower. The second border shape is a Singe edge with a slope $m = 1.5$ close to 1, denoted by SE. The bottommost node is placed so that the map has the shape of a right triangle, see Figure 5.11 (b). The third border TE consists of Two edges with a positive and a negative slope, both strongly slant so that the map has the shape of an isosceles triangle , see 5.11 (c). The fourth border CIR is composed of five edges so that the map has approximately a Semicircular shape, see 5.11 (d). The last shape of border depends on the positions of the points. Given a position of n points, the last border corresponds to the right side of the convex hull of the points, after being slightly shifted to the right so that no point touches the border, see Figure 5.11 (e). This border is denoted by CH.



(a) Border VB (b) Border SE (c) Border TE (d) Border CIR (e) Border CH

Figure 5.11: The different shapes of the border. The red strokes correspond to the convex boundary.

In addition to a general form of the border, we also have to choose the height of the labels. If the height of the label is too big, then the total size of the labels will be close to the height of the boundary. Thus, it would not be interesting to use sliding labels, since such a problem can be more efficiently solved by algorithms for fixed label slots. On the other hand, if the height of the labels is to small, there will be a lot of clusters, which is not interesting for experimental results because the optimal position of the labels will rarely intersect, and thus the minimal labeling will be computed in linear time [1]. We arbitrarily choose a height h of the labels equals to $\frac{3}{4}H$, where H is the height of the boundary.

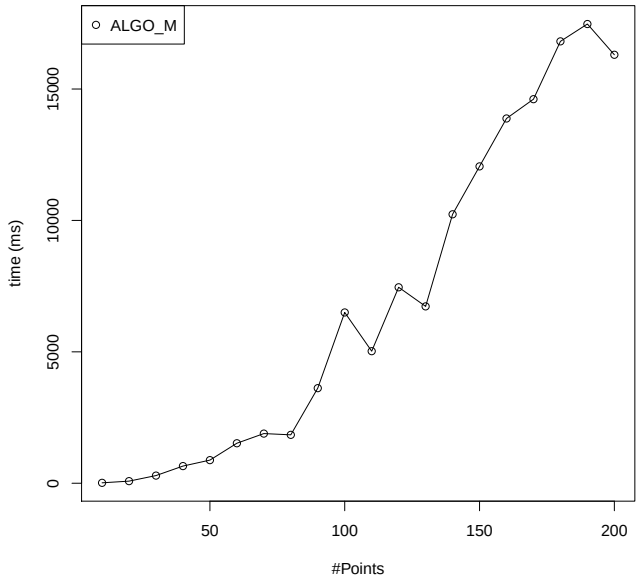
Number of Points. In a map, we generally compute labeling for a small number of points. In the previous section, we saw boundary labelings for a map of Germany and a map of Italy. The objective was to annotate the name of each region. In both case there were not more than 20 labels. In real maps, to have a satisfying readability, the number of label should not exceed a few hundred. In our experiments, we compute labeling with at most 200 points. Moreover, the points are randomly chosen depending on a uniform function.

Results. The results in this section are based on 1000 randomly chosen instances of the points.

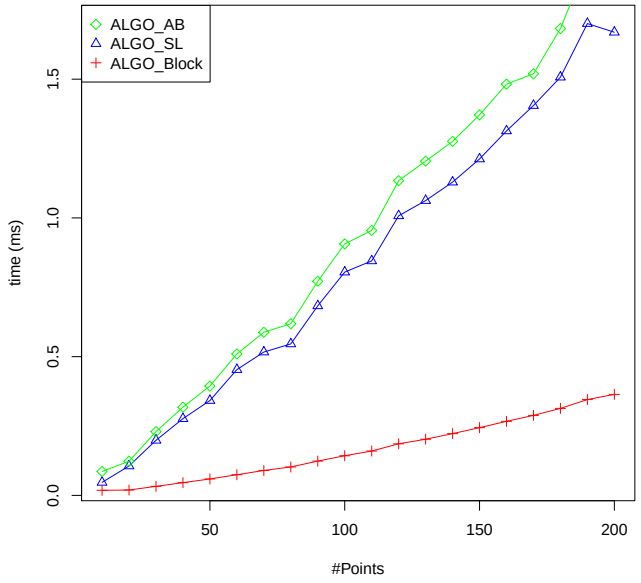
We first study the runtime when the border is vertical. This first case is studied to points out the time that ALGO_M takes by computing $O(n) \cdot O(n^2)$ matchings. Figure 5.12 presents the results of our experiments:

As expected, ALGO_SL and ALGO_AB have the same run time. ALGO_Block is slightly faster since it computes an optimal cluster position in constant time without making a sweep-line. On the other hand, ALGO_M takes nearly a second to compute a labeling of 50 points. For 200, points, the run time is $15s$, which is 10.000 times slower than the

[1] Since there will be few calls to the algorithm computing minimal cluster labeling.



(a) ALGO_M



(b) Others algorithms

Figure 5.12: Run time of the four algorithms when the border is vertical. Figure (*a*) presents the runtime for ALGO_M which is 10.000 times slower than the others.

other algorithms. Moreover, for the other border shapes, ALGO_SL takes even more time to compute an optimal labeling. This comes from the fact that for a vertical border, the points are uniformly distributed, which reduce the chances for the optimal position of two labels to overlap. In a map with 200 points and the Two-edges border presented above, ALGO_M takes approximately $50s$. We deduce that it is worth first computing a labeling with Algorithm 10 and avoid as much as possible the matching algorithm.

Now, we present the average run times for each of the three algorithms ALGO_SL, ALGO_AB and ALGO_Block for the different shapes of the border.

Figure 5.13 shows the average run times of ALGO_SL for the different border shapes.

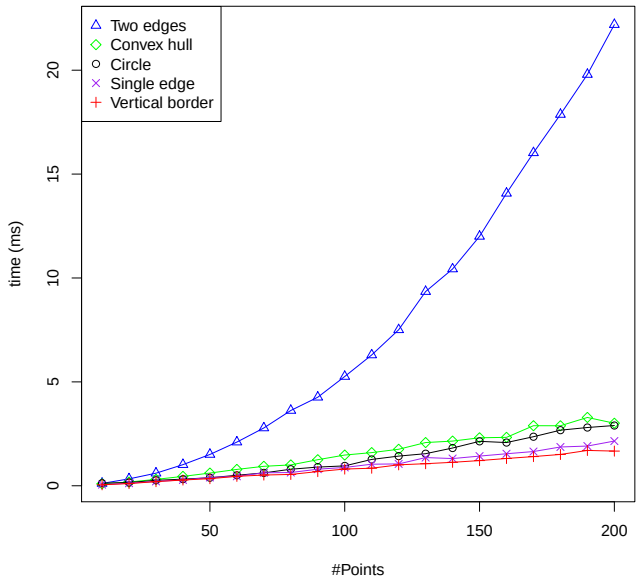
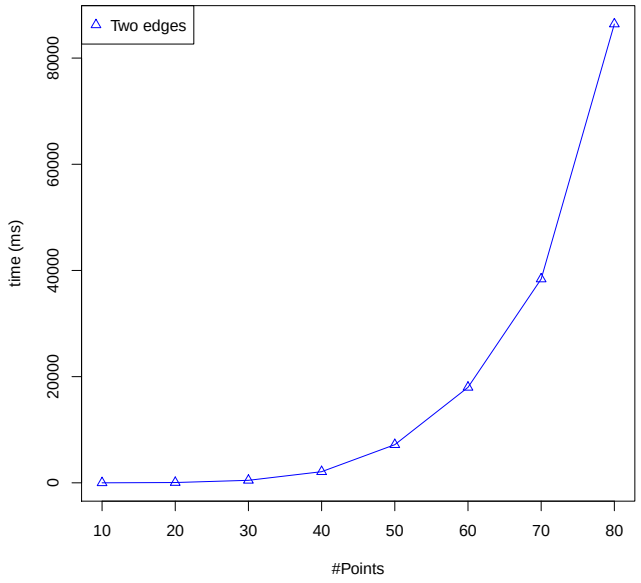


Figure 5.13: Run time of ALGO_SL for different border shapes.

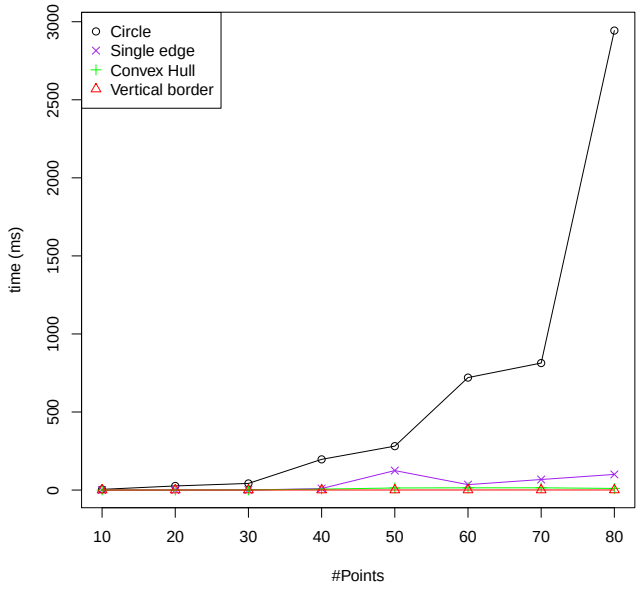
The algorithm takes much more time for the boundary containing two edges. It is probably due to the fact that the shape of the map is such that every point is close to the border. In Figure 5.11, we can see that the map where the points can be placed is really narrow. For others border shapes, ALGO_SL takes approximatively the same amount of time to compute an optimal labeling.

We study now the results of ALGO_AB for each of the five border shapes, presented in Figure 5.14. As for ALGO_SL, the run time of ALGO_AB for the border with two edges takes a lot more time than for the other border shapes. For 80 points, the run time is approximately $80s$. Therefore, ALGO_SL mostly computes labeling with several shifted labels for this shape of border, which makes ALGO_AB much less efficient. For a border with a circular shape, the run time also increases rapidly. For 80 points, it takes 3 seconds to compute an optimal labeling. Further computations shows that for 200 points, the run time exceeds $5s$. ALGO_AB takes much less time than ALGO_M, but still significantly more time than ALGO_SL, see Figure 5.13. We also see here that first calling the sweep-line algorithm in Algorithm 14 reduce immensely the run time. For the convex hull shape, further computations shows that it takes around $25ms$ to compute an optimal WAB-labeling with ALGO_AB. The speed of ALGO_AB for this border can be explained with the same argumentation as for a vertical border. Since most points are placed far away to the left of the boundary, most labels in the labeling computed by ALGO_SL are not be shifted. We deduce that, for most maps, Algorithm 14 computes an optimal cluster labeling in a reasonably small run time, but for a few maps, the algorithm must repetitively call Algorithm 9 to compute a minimal cluster labeling, and consequently the run time increase immensely.

We finally present in Figure 5.15 the average run times of ALGO_Block:



(a) Border with two edges



(b) Others border shapes

Figure 5.14: Run time of ALGO_AB for different border shapes.

Just as expected, the algorithm is incredibly fast for every border shape, since the algorithm computes in constant time an optimal cluster position.

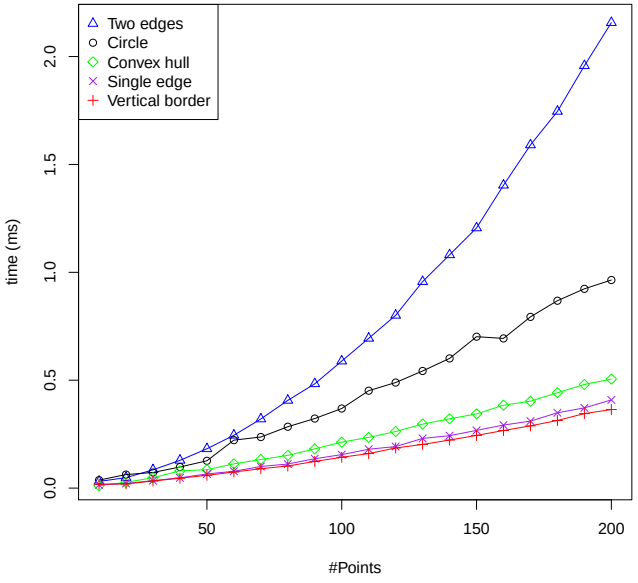


Figure 5.15: Run time of ALGO_Block for different border shapes.

6. Conclusion

In this thesis, we studied different models of map labeling with a convex boundary. The motivation of this work was to improve the visual quality of a labeling when the shape of the map border is far from being vertical. A lot of research has already be done for one-sided boundary labeling, and one of the models introduced by Bekos et al. [BKSW07] is to create a labeling where the points and the labels are connected with non intersecting po-leaders and so that the total leader lengths is minimal. When the right side of the boundary is not vertical, it is possible to place the anchor of the labels to the left of a point, which should not happen in the proposed model. Because of this fact, the model used until now is not complete anymore, and further constraints must be added. In this work, we assumed that the border is composed exclusively of edges with a big slope. Otherwise, there would be several locally optimal positions for a label, and thus the problem would be similar to a multi-sided labeling problem. Moreover, we concentrated our search on a right-sided labeling problem, and considered that an edge with small slope corresponds to the top-side or the bottom-side of the convex border. We studied three different ways to adapt the existent model.

In the first model, we allowed labels to be placed away to the right of from the boundary. In this case, the *shifted* label is right below the point it is connected to. We called this model *weakly-aligned boundary* labeling model.

The second model requires label to touch the border and allows points and labels to be connected with op-leaders, when the point is situated to the right of the label. This second labeling model is denoted by *strictly-aligned boundary* labeling.

These two models produce good-quality labelings, but the presence of op-leaders and shifted labels makes the computation of an optimal labeling more difficult. As a result the worst case time complexity increase immensely. However, a first fast solution can also been produced, and a simple criteria can tell whether the produced labeling is optimal or not. Experiments showed that including this fast algorithm in our algorithm reduces the average run time drastically. A further problem is that the Upward-downward algorithm used for a vertical or with a convex border consist of a single edge become suboptimal in the case of a border composed of several edges, which add a worst case time complexity factor exponential depending on the number of edges. It is still not clear how much this adaptation can increase the run time, and if this factor is really attained in worst case.

In a last model, we required that the clusters of labels in a labeling must be vertically aligned in order to prevent the difficulty encountered in the two previous models. To

make the visual quality of the labelings produced with this new model, we added a strong constraint of *local optimality*. With this last model, the labeling are produced as fast as for a vertical boundary, and the position of the labels also take into account the shape of the border. However, when there are too much labels with a too big height, the produced labeling may contain a single cluster and the position of the labels would then not follow the shape of the border into account. The experimental results showed that solutions for realistically-sized instances are computed instantaneously.

Computing two-sided labelings with minimal leader length and no crossing is a further challenging problem. It has been studied by Bekos et al. [BKSW07] for fixed labels positions, but little has been done yet for sliding labels, that is, for labels which can have any position as long as they do not intersect.

Moreover, it would be interesting to adapt the algorithms for vertical boundary in the case of dynamic maps, where it is possible for the user to zoom in and out and make the position of the labels evolve continuously.

Finally, changing slightly the shape of the leader so that leader are not too close to each others and far enough to the points would be a thrilling research problem. A labeling would then be produced by a model either containing a further constraint on the leader, or including this constraint in the badness function.

Acknowledgment

I want to thank my advisors Martin Nöllenburg and Andreas Gemsa for giving me the opportunity to work on this topic, for having always time and patience for questions and for provinding me valuable advises in our discussions.

Bibliography

[BHKN09] M. Benkert, H. Haverkort, M. Kroll, and M. Nöllenburg, "Algorithms for multi-criteria boundary labeling," *Journal of Graph Algorithms and Applications*, vol. 13, pp. 289–317, 2009.

[BKNS09] M. A. Bekos, M. Kaufmann, M. Nöllenburg, and A. Symvonis, "Boundary labeling with octilinear leaders," *Algorithmica*, vol. 57, pp. 436–461, 2009.

[BKPS06] M. A. Bekos., M. Kaufmann, K. Potika, and A. Symvonis, "Multi-stacks boundary labeling problems," *Springer Heidelberg*, vol. 4337, pp. 81–92, 2006.

[BKPS08] ——, "Area-feature boundary labeling," *The Computer journal*, 2008.

[BKSW07] M. A. Bekos, M. Kaufmann, A. Symvonis, and A. Wolff, "Boundary labeling: Models and efficient algorithms for rectangular maps," *CGTA*, vol. 36, pp. 215–236, 2007.

[Cca99] B. Chazelle and . co authors, "The computationary geometry impact task force report," *Advances in Discrete and Computational Geometry*, vol. 23, American Mathematical Society, Providence, RI, pp. 407–463, 1999.

[CR99] V. Chandru and M. R. Rao, "Integer programming," vol. 32, pp. 31/1–32/45, 1999.

[GHN11] A. Gemsa, J.-H. Haunert, and M. Nöllenburg, "Boundary labeling algorithms for panorama images," *ACM GIS 2011*, 2011.

[GJ79] M. R. Garey and D. S. Johnson, "Computers and intractability: A guide of the theory of NP-completeness," *Freeman*, 1979.

[Imh75] E. Imhof, "Positioning names on maps," *The American Cartographer*, vol. 2, pp. 128–144, 1975.

[Kar84] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, pp. 373–395, 1984.

[Kau09] M. Kaufmann, "On map labeling with leaders," *Springer*, pp. 290–304, 2009.

[Kuh55] H. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistic Quartery*, vol. 2, pp. 83–97, 1955.

[Law76] E. L. Lawler, "Combinatorial optimization: Networks and matroids," *New York: Holt, Rinehart & Winston*, 1976.

[LK08] C.-C. Lin and H.-J. Kao, "Many-to-one boundary labeling," *Journal of Graph Algorithms and Applications (JGAA)*, vol. 12(3), pp. 319–356, 2008.

[NPS10] M. Nöllenburg, V. Polishchuk, and M. Sysikaski, "Dynamic one-sided boundary labeling," *ACM GIS'10*, 2010.

[Vai89] P. M. Vaidya, "Geometry helps in matchings," *SIAM Journal on Computing*, 1989.

[vB10] L. Čmolík and J. Bittner, "Layout-aware optimization for interactive labeling of 3d models," *Computers and Graphics*, vol. 34, pp. 378–387, 2010.

[Wag93] F. Wagner, "Approximate Map Labeling is in $\Omega(nlogn)$. Technical Report b 96-98," *Fachbereich Mathematik und Informatik, Freie Universität Berlin*, 1993.

[Wol96] A. Wolff, "The map labeling bibliography," 1996. [Online]. Available: http://i11www.ira.uka.de/map-labeling/bibliography/

[Yoe72] P. Yoeli, "The logic of automated map lettering." *The Cartographic Journal*, vol. 9, pp. 99–108, 1972.