

Map Labeling Problems

by

Claudia C. Iturriaga-Velázquez

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Doctor of Philosophy

in

Computer Science

Waterloo, Ontario, Canada, 1999

©Claudia C. Iturriaga-Velázquez 1999



**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-44766-9

Canada

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

One of the most challenging tasks in cartography is the **labeling** of maps—attaching text to geographic features. Many of the issues become simpler when the features are points, for example cities on a large-scale map, because we expect the text to be placed horizontally and close to the associated point. We want labels that do not overlap and are large enough to be readable. Even simple formulations of this problem are NP-complete. One such formulation is the *point-feature label placement problem*: given a set of points in the plane, and an axis-parallel rectangular label associated with each point, place each label with one corner at the associated point such that no two labels overlap. This problem is known to be NP-complete.

Modeling each label as a *fixed* rectangle in this way is quite limiting. Researchers have considered approximation algorithms where each label can be *scaled*.

In this thesis, we propose an alternative formulation to the map labeling problem. We introduce the use of *elastic labels*, where each label is a rectangle with fixed area, but varying in height and width. Then, we define the *elastic labeling problem* as determining the choice of height and width of each label, and the corner of the label to place at the associated point, so that no two labels overlap. This problem is useful when the goal of placing a label at a given point is to associate some text, consisting of more than one word, with the point. In this case we can write the specified text inside the label using one, two, or more rows, as long as the label is placed at the specified point.

The elastic labeling problem is an NP-hard problem, since it generalizes the point-feature label placement problem. That is, the problem is NP-hard even when there is no elasticity just because of the choice of the corners. We show that the problem remains NP-hard when we have elasticity but no choice about which corner

of the label to use. We call this the *one-corner elastic labeling problem*.

Thus, we consider a problem with more constraints, yet of practical use. We require that the points lie on the boundary of a rectangular map. This *rectangle perimeter labeling problem* arises, for example, when the perimeter of the map is labeled with information about objects that lie beyond the boundary of the map, e.g. where the roads lead to. This problem is likely to be relevant in Geographical Information Systems (GIS) as maps are displayed dynamically on a computer screen using clipping, panning, and zooming. It is also applicable for labeling the coordinate axes in graphs such as charts or histograms.

Our main result is a polynomial time algorithm for the rectangle perimeter labeling problem. To solve the rectangle perimeter labeling problem, we look at two problems. The first problem is when the points lie on two adjacent sides of the rectangular map, the *two-axis labeling problem*. The second problem is when the points lie on two opposite sides of the rectangular map, the *two-parallel lines labeling problem*.

In this thesis, we also study another variant of the point-feature label placement problem: the *sliding labels problem*. In this problem each label has fixed dimension but any boundary point (not just a corner) of the rectangular label can be placed at the associated point in the plane. This problem is known to be NP-hard. We show that it remains NP-hard when points can only be placed on the left or right sides of the boundary of their associated labels—we call this the *left-right sliding labels problem*.

As well we give algorithms for special cases of the elastic labeling problem and the sliding labels problem. Finally, we study approximation algorithms for the point-feature label placement problem.

Acknowledgments

I wish to express my sincere thanks to my supervisor, Professor Anna Lubiw, for her guidance and patient supervision. She made the successful completion of this work possible. Thanks to her for teaching me how to refine my ideas and how to do research.

I am grateful to the members of my thesis committee Len Guelke, Joe Marks, Ian Munro, and Naomi Nishimura for their thorough reading, valuable suggestions, comments, and motivating questions for future work.

I would like to thank my friends Atousa Mohammadi, Alice Li, Mike Vadera, Claudia Rodríguez, and Daniel M. Germán for cheering me up and sharing laughs as well as difficult times with me. Of the many friends I have made in Waterloo, I would like to thank Gudrun Wessel, Soeren Peik, Kathy and James Hodge, Janaki Bandara, Gina Holguin, Arun Marathe, Peter Vanderheyden, Erik and Marty Demaine, Wendy Rush, Marcelino Madrigal, Francisco Zaragoza, and María and Torsten Nelson. Additional thanks to Alice, Mike and Kathy as well for their careful reading of early drafts of some chapters. Thanks go as well to the López-Ortiz family. Thanks to Jennifer Keir, who took care of the submission of this thesis for my defense and to Daniel M. Germán, who submitted the final version.

I wish to thank my parents, Carmen and Sergio, my brother Miguel, and my sisters Celia and Juanita for many nice memories through my life, and especially for their unconditional love and support that helped me out during the final push for this work. Special thanks go to my uncle Jesús Velázquez for his great emotional support during the last push of this work and for “being there” in this crucial time of my life.

No words can express my gratitude to Alex López-Ortiz for his constant encour-

agement, support, advise, suggestions, discussions, for “being there” many times when I needed him, for his love, for the time that we shared together, for many happy times together and for many, many things that we learned together.

Finally, the financial support from the National University of Mexico (UNAM) recommended by the Institute of Mathematics, and the Department of Computer Science at the University of Waterloo is gratefully acknowledged.

*To Alex and Claudia,
and many dreams to come.*

Contents

1	Introduction	1
1.1	Motivation	1
1.2	The Art of Lettering Maps	2
1.3	The Label Placement Problem	6
1.3.1	Point-Feature Label Placement Problem	7
1.3.2	Other Variations of the Label Placement Problem	12
1.4	Overview	13
2	Definitions and Notation	16
2.1	Elastic Labeling Problem	16
2.1.1	Motivation	16
2.1.2	Definitions and Notation	27
2.2	Sliding Labels Problem	32
2.2.1	Motivation and Definitions	32

3	The Two-Axis Labeling Problem	35
3.1	Definitions and Preliminaries	35
3.2	Structure of a Good Realization	37
3.3	Algorithm for the Two-Axis Labeling Problem	40
3.4	Two-axis algorithm.	42
3.5	Hyperbolas	48
3.5.1	Elastic Rectangles in the Same Axis	48
3.5.2	Elastic Rectangles in Opposite Axes	51
3.6	Discrete Two-Axis Labeling Problem	54
4	Two-Parallel Lines Labeling Problem	59
4.1	Definitions and Preliminaries	59
4.2	An Algorithm for the Two-Parallel Lines Case	61
4.2.1	Fixing a Rectangle	65
4.2.2	Algorithm	75
4.2.3	Hyperbolas	77
4.2.3.1	Procedure using-four-hyperbolas	82
4.3	Discrete Two-Parallel Lines Labeling Problem	83
4.4	One-Line Labeling Problem	86
5	The Rectangle Perimeter Case	89
5.1	Definitions and Preliminaries	89

5.2	Boundary Rectangle Decomposition	91
5.3	Existence of a Corridor	96
5.4	Corridor Partition Realization Problem	102
5.5	Algorithm	106
5.6	Discrete Rectangle Perimeter Labeling Problem	107
5.7	The Exterior Rectangle Perimeter Case	108
5.7.1	The Exterior Rectangle Labeling Problem with Minimum Border Band	111
5.8	Rectangle Perimeter Labeling Problem with an Obstacle	114
5.9	Conclusions	117
6	Variants Using Elastic and Sliding Labels	118
6.1	Same-Corner Elastic Labeling Problem	118
6.2	Same-Side Sliding Labels Problem	125
7	Approximation Algorithms for Labeling	128
7.1	Introduction	128
7.2	Squares of Different Sizes	133
7.3	Rectangles of Different Heights	136
8	NP-hardness	141
8.1	One-Corner Elastic Labeling Problem	141
8.2	Sliding Labels Problem	154

8.2.1	Left-Right Sliding Labels Problem	154
8.3	Three-Position Point-Feature Labeling Problem	162
9	Conclusions	170
	Bibliography	173

List of Figures

1.1	Labeling cities.	3
1.2	Labeling a river.	4
1.3	Labeling a region.	4
1.4	Poor labeling produced by labels overlapping.	5
1.5	Poor labeling produced by illegible text.	5
1.6	Good Labeling.	5
1.7	Poor labeling.	5
1.8	An instance of the point-feature label placement problem.	10
2.1	A Labeling Map.	17
2.2	A diagram with text labels.	18
2.3	Using an elastic label to set a text label.	19
2.4	London Underground.	22
2.5	Yorkville, Toronto.	23
2.6	A solution for an instance of the rectangle perimeter labeling problem.	24
2.7	A solution for an instance of the two-axis labeling problem.	25

2.8	A solution for an instance of the two-parallel lines labeling problem.	25
2.9	A decomposition.	26
2.10	An elastic rectangle $\mathcal{E}$ with $Q(\mathcal{E}) = \{bl\}$	27
2.11	An elastic rectangle $\mathcal{E}$ with $Q(\mathcal{E}) = \{bl, tl, tr, br\}$	28
2.12	An instance of the elastic labeling problem.	29
2.13	A bad realization for the instance of Figure 2.12.	30
2.14	Good realization obtained by using elasticity.	31
2.15	Good realization obtained by changing corner selection.	31
2.16	Sliding Labels.	32
2.17	A good realization for an instance of the sliding labels problem. . .	33
3.1	A good realization for the two-axis labeling case.	36
3.2	Elastic rectangles facing each other.	37
3.3	Good clear realizations.	38
3.4	Peeling way rectangles in a clear realization.	39
3.5	Clear decomposition of a good realization.	40
3.6	A good realization with a non-unique minimum height and width. .	42
3.7	Adding elastic rectangle $\mathcal{B}_2$ to a right-clear realization.	43
3.8	Case I.1.	44
3.9	Case I.2.	45
3.10	Elastic rectangles whose hyperbola segments intersect.	50
3.11	Realizations with boundary value $B_i^{h_{\max}}$	52

3.12	Elastic rectangles in opposite axes whose hyperbola segments intersect.	53
3.13	Elastic rectangles in opposite axes whose hyperbola segments do not intersect.	54
3.14	Good realizations for instances of the discrete two-axis labeling problem.	58
4.1	A good realization for an instance of the two-parallel lines labeling problem.	60
4.2	Rectangles facing each other.	61
4.3	Smallest-moves-up ordering.	62
4.4	C-shape.	64
4.5	Adding anchor q to the topmost C -shape.	65
4.6	Anchor u leaves the topmost C -shape.	66
4.7	Fixing an elastic rectangle to a realization lying below t	68
4.8	Fixing $\mathcal{U}$ for Case 1.	69
4.9	Fixing $\mathcal{U}$ and $\mathcal{S}$ for Case 2.	70
4.10	Fixing $\mathcal{U}$ for Case 3.	71
4.11	Fixing $\mathcal{U}$ and $\mathcal{S}$ for Case 4.	73
4.12	A realization for $\mathcal{U}$ and $\mathcal{S}$ of minimum total width.	74
4.13	Cases of a realization when r , s , and t are in the topmost C -shape.	75
4.14	Scenarios for a good realization.	78
4.15	a) A realization having $\mathcal{U}$ and $\mathcal{S}$ of minimum total width, and $\mathcal{U}$, $\mathcal{T}$ of minimum total height. b) A good realization for a).	80

4.16	An example of a good realization for the exterior case.	86
5.1	A decomposition.	91
5.2	Examples of corner blocks shown in dashed bold rectangles.	92
5.3	Illegal corner block.	92
5.4	Touching corner blocks shown in dashed bold rectangles.	93
5.5	A horizontal corridor.	94
5.6	Free jagged horizontal lines.	96
5.7	Building a free jagged line.	98
5.8	Free horizontal jagged lines.	99
5.9	Corner blocks covering the bottom side of P cannot be built.	101
5.10	Building a Corridor when a Horizontal Corridor does not exist.	102
5.11	Building an elastic block.	104
5.12	Labels in the exterior of a rectangular image.	109
5.13	An instance of the exterior rectangle perimeter labeling problem.	110
5.14	Realizations on the corners.	111
5.15	Close-to-corner rectangles.	113
5.16	Rectangle perimeter labeling problem with an obstacle.	116
6.1	Current rectangle intersecting active rectangles.	122
6.2	An instance of the same-side sliding labels problem with anchors not in general position.	127

7.1	A maximal independent set for squares (indicated in bold squares).	135
8.1	Basic building block.	143
8.2	Pipes.	143
8.3	Variable setting.	144
8.4	Clause gadget.	145
8.5	Layout of variable, clause, and pipe gadgets.	146
8.6	Connection between pipes.	147
8.7	Vertical part of the crossing gadget.	149
8.8	Horizontal part of the crossing gadget.	150
8.9	Crossing gadget.	151
8.10	Vertical flow only.	152
8.11	Vertical and horizontal flow.	152
8.12	Horizontal flow only.	153
8.13	Neither vertical nor horizontal flow.	153
8.14	Basic blocks.	155
8.15	Pipes.	156
8.16	Variable gadget.	157
8.17	Clause gadget.	157
8.18	Clause gadget forcing flow into pipe V_3	159
8.19	Connection of pipes.	160
8.20	Crossing gadget.	161

8.21	Candidate positions for a label.	163
8.22	Pipes.	163
8.23	Variable gadget set to False.	164
8.24	Variable gadget set to False.	165
8.25	Variable gadget set to True.	166
8.26	Vertical pipes connected to a variable gadget set to True.	166
8.27	Vertical pipes connected to a variable gadget set to False.	167
8.28	Clause gadget.	167
8.29	Clause gadget.	168
8.30	Clause gadget.	168
8.31	Crossing Gadgets.	169

Chapter 1

Introduction

A key factor in a map's ultimate effectiveness as a communication medium is the labeling of the features.

Doerschler and Freeman,
Comm. ACM, 1992.

1.1 Motivation

Most maps or charts are made of drawings together with textual information that one wants to convey, for example, names of employees in an organizational chart, names of countries or cities on maps, relevant data in histograms, or names of servers in a computer network.

In this thesis, we focus on the problem of attaching textual information to maps or diagrams. In particular, we emphasize maps, though the techniques can be used for any diagram.

One of the fundamental tasks to achieve in cartography is the placement of information on a map. Textual information is usually placed with labels that are located close to the features that they are tagging. Labels in a map usually describe names of rivers, countries and cities. Typically, labeling geographical features on a map consists of the following tasks among others:

- determine the size of a label,
- find a set of candidate positions for a label,
- choose a position for each label from the set of candidate positions,
- select the set of labels to be shown.

In this thesis we concentrate on the third problem above, namely, how to choose a position for each label from its set of candidate positions. However, to provide sufficient background, we briefly present some aspects of the other problems (Section 1.2), and then we survey the work done for the third problem (Section 1.3).

1.2 The Art of Lettering Maps

Map lettering provides communication between the map user and the cartographer. Lettering is an art in cartography. It conveys information with characteristics such as style, size, and placement. For example, with respect to *style* sometimes italic fonts are used to label hydrographic features such as oceans, seas, bays, lakes and rivers. The *size* of a label helps us to identify important features on a map, for example, having bigger labels for the more important cities. The *placement* of the label helps us to identify which feature is being described.

Most of the rules for such characteristics were transmitted from generation to generation. There are some attempts to write these rules down in *Lettering on Topographic Maps* [IGN34] and in an essay by Bonacker [Bon57]. In order to formalize the lettering characteristics, Imhof [Imh75] described æsthetic criteria to characterize good quality map lettering. These criteria give a set of rules of how a region should be labeled to agree with its geographical characteristics, like if the feature is a city, river or mountain. We discuss some of these criteria below.

Labeling features are classified as:

1. Position designation or names for point-like objects and concepts like cities, drill-sites, hospitals, or post offices. These labels should be next to, over, or under the object, without breaks. They are usually horizontal (see Figure 1.1).

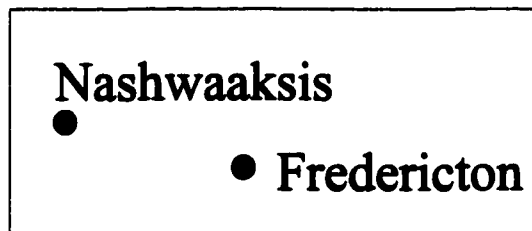


Figure 1.1: Labeling cities.

2. Linear designations such as rivers, creeks, paths, streets, railroads, air lines. These labels are placed along the lines to which they refer (see Figure 1.2).
3. Area designations such as woods, fields, mountains, countries, states, lakes. These labels require bending and spreading a name so that it is stretched as much as possible across the designated area (see Figure 1.3).



Figure 1.2: Labeling a river.

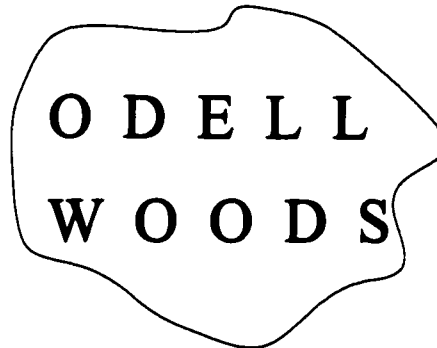


Figure 1.3: Labeling a region.

We concentrate on problems of the first type of designation. An interesting problem arises when the map contains thousands of nodes that are too close, which makes the labels on the nodes hard to read. It is difficult to place the labels in such a way that they do not overlap and will be legible. By legible, we mean that the size of the label can be read easily without magnification.

For example, Figures 1.4 and 1.5 depict a bad placement of labels in a map. Figure 1.4 has a poor labeling since it has labels that overlap. Figure 1.5 has a poor labeling since it has labels that are too small to be read. A good labeling of this map is shown in Figure 1.6.

Another desired property in good labeling is to avoid ambiguity. Labels are usually required to be in close proximity to—but not overlapping—the objects they describe. In principle we could select the labeling shown in Figure 1.7, however in this case, it is unclear which city or dot is Waterloo and which is Kitchener.

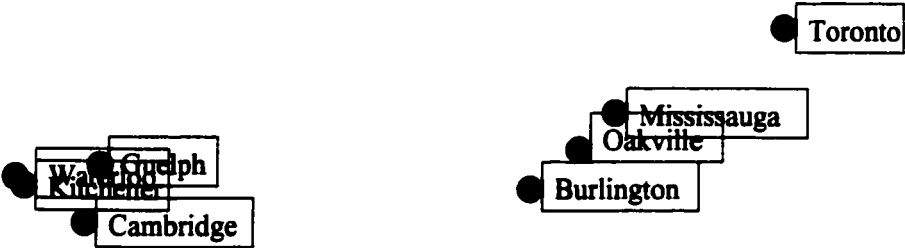


Figure 1.4: Poor labeling produced by labels overlapping.

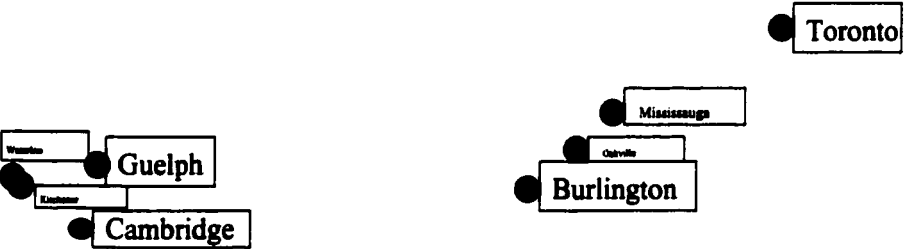


Figure 1.5: Poor labeling produced by illegible text.

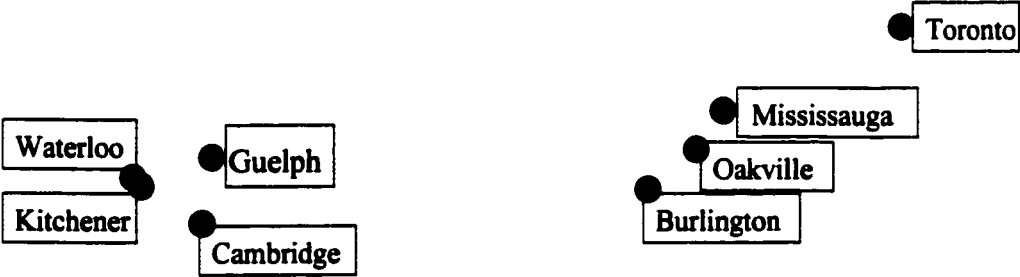


Figure 1.6: Good Labeling.



Figure 1.7: Poor labeling.

Beus and D. Crockeit [BCr94] commented that in the production of maps, computer technology excels at producing graphics as good as that of maps produced using traditional means; but they argue that technology cannot replace the experience of editors and that the craft of map lettering is still an indispensable ingredient of cartography. Map lettering is a task usually performed by hand by expert cartographers [Woo87, Tyn92, RMG95, Den96]. Although computers have brought efficiency to the placement of labels, sometimes human intervention is required to improve the quality of the labeling. However, Geographical Information Systems (GIS) and chart generation software require automatic placement of labels. There is also a need for labeling a huge number of maps, as pointed out by Ebinger and Goulette [EGo90]. They developed a heuristic to label an estimated 1.3 million different map sheets for the 1990 Decennial Census of the USA.

Thus, we need algorithms that produce good quality labeling without human intervention. Consequently, the area has been the subject of intense research [CMS95, DMM97, ECM97, FWa91, Hir82, KSW98, KMP93, MSh91, RMG95, Wag94, WWo97, KSW98, WWo98].

1.3 The Label Placement Problem

In this thesis, we focus on the placement of information on a map or a chart. In particular, we study the case when we want to label points. First we define the label placement problem and then we concentrate on labeling points.

Definition 1.1 *Given a set of n objects in a map, each object associated with a label, the **label placement problem** is to find a position of each label close to the object to which it refers such that no two labels overlap. In this case, close may*

mean touching the object or may mean within some given distance.

The label placement problem has been identified as a key problem and for research by the ACM Computational Geometry Impact Task Force [Cha96]:

Label placement is a source of many geometric optimization problems. Even when labels are to be placed in fixed positions relative to point features, these are typically NP-hard. Since maps do need labels, heuristics and efficient algorithms for identifying constraints and possible positions are needed.

There is a good bibliography on map labeling collected by Wolff and Strijk [WSt96].

Most of the work done in the label placement problem assumes that the features to label are points. We survey the work done for labeling point features in the following section.

1.3.1 Point-Feature Label Placement Problem

Labeling point features has received most of the attention of the research addressing the label placement problem. Some examples of point features to be labeled are cities, drill-sites, hospitals, electrical power stations and post offices. In statistics, we can label scatterplots, which are x - y graphs of data on which the x -coordinate corresponds to the value of one variable and the y -coordinate to the value of the other variable [ISr89]. Scatterplots provide a good feel for how the two variables are co-related. They are also useful for highlighting aberrant data.

It is easy to see that the problem of labeling point features becomes difficult when the map contains thousands of points that are close together. In this case,

the difficulty lies in placing the labels so that they do not overlap and are easy to read without magnification.

Yoeli [Yoe72] presented the first attempt to automate labeling. Most of the first heuristics in the area [AFr84, FrA87, AIK89, Jon89, CJo90, DFr92] try all possible positions for a label, thus in terms of time they are not very efficient.

Heuristics have been developed for the case when the point can be in any part of the boundary of its label. For this case Christensen et al. [CMS95] presented a good empirical study of these heuristics and they classify them as follows:

- physical relaxation: labels are moved in response to virtual forces generated by label overlaps [Hir82].
- 0-1 integer programming: scores are associated to each candidate position of a label, and are refined iteratively to reflect the relative desirability of the different positions [Zor86, Zor90].
- gradient descent: a random labeling is generated and improved monotonically by considering all candidate positions for each label and moving the label that most improves the quality of the whole labeling [CMS93].
- simulated annealing: this generalizes the gradient descent method and allows moves of a label that worsen the quality of the labeling in order to escape local optimal labeling [CMS95].

Christensen et al. [CMS95] presented experiments where the method of using simulated annealing betters all the other above methods. Most of the above heuristics are for the case when the set of candidate positions is discrete, except for the case of physical relaxation. In this case the point may be located in any part of

the boundary of its label and the corresponding label placement problem for this version is known as the *sliding labels problem*. Van Kreveld et al. [KSW98] studied this problem as well and gave an approximation algorithm for it. We talk more about their results in Chapter 7.

Now, we proceed to present the main variant of several studies for the labeling of points, and the one which we focus our attention in this thesis. An important and traditional formulation of the label placement problem is the *point-feature label placement problem*.

Definition 1.2 [FWa91, KIm88, KRa92, MSh91]: *Given a set of points on the plane, each point associated with an axis-parallel rectangular label, the **point-feature label placement problem** is to place the labels so that they do not overlap, and so that each label has one of its corners at its associated point.*

Figure 1.8 depicts an instance of the point-feature label placement problem, showing at each point the four candidate positions of its associated label, having in bold the rectangles chosen to form a good labeling of the instance. The point-feature label placement problem is known to be NP-complete [FWa91, KIm88, KRa92, MSh91] and heuristics have been proposed for it [CMS95, FWa91, WWo95].

Kučera et al. [KMP93] gave two exact algorithms to solve the point-feature label placement problem. The running times of these algorithms are $O(4^{\sqrt{n}})$ and $O(4^{\sqrt{n \log n}})$. These algorithms can only be used for small instances of the problem. For usual real world data, where the size of the instances is very large, these algorithms are not practical. Then, heuristics or variants of the problem are a good approach to study this problem.

One variant of the problem is when each point allows only two possible candidate positions for a label. Formann and Wagner [FWa91] solve this problem

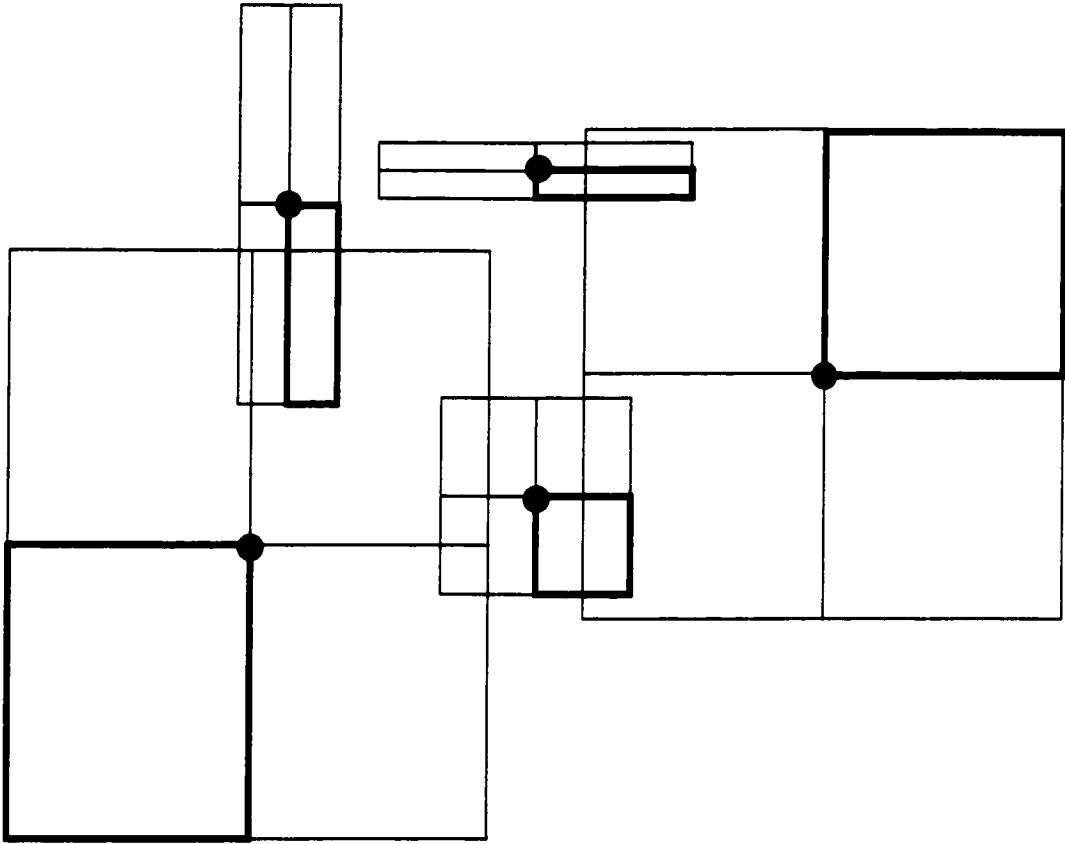


Figure 1.8: An instance of the point-feature label placement problem.

by formulating a 2-SAT instance of it. In this case we can solve the problem in $O(n \log^2 n)$ time.

Formann and Wagner [FWa91] presented an approximation algorithm (see Chapter 7 for definition) for finding the maximum size or scaling of the labels that can be used, having all points labeled with same size labels. They proved that, unless $P = NP$, no polynomial-time approximation algorithm can do better than 50% of the optimal size of the labels. They gave an $O(n \log n)$ time approximation algorithm for the optimal size of the labels. Wagner [Wag94] proved later that this time bound is tight by proving that any approximation algorithm that achieves a

labeling having at least labels of half of the size of the optimal takes $\Omega(n \log n)$ time. Wagner and Wolff presented in [WWo97] a heuristic that has shown better results than the approximation algorithm of Formann and Wagner. Christensen et al. [CMS97] compared the algorithm of Wagner and Wolff with their own which uses simulated annealing [CMS95]. They found that the algorithm of Wagner and Wolff was better than the one using simulated annealing. The results were favorable as well for the algorithm using simulated annealing since its results were close to the other algorithm. As we discussed above this is important since experiments by Christensen et al. [CMS95] have reported that simulated annealing overcomes other algorithms for a more general variant of the point-feature label placement problem.

In the literature we can also find approximation algorithms [HMR95, AKS98, KSW98] (see Chapter 7 for definition) to label the maximum number of points possible, for the case where all labels have the same height. There are no known approximation algorithms for the general problem where labels may have different heights and widths.

With the advent of Geographical Information Systems (GIS) maps now are more dynamic in nature. Consider for example, a weather map or a traffic map where the information changes constantly. Arikawa et al. [AKK94] proposed “dynamic maps”; these maps provide the users with more flexible, specialized and reliable information than paper maps. Some of the characteristics of these maps are that they allow the user to produce maps which consist only of the data that is of interest to the user.

Using GIS maps, labels can include more information about objects that the user has selected. Keeping this possibility in mind we introduce the notion of *elastic labels* in this thesis (see Section 1.4).

1.3.2 Other Variations of the Label Placement Problem

The label placement problem in general refers to the labeling of any object. As we discussed in section 1.2 there are different features besides points that are also labeled. In this section, we talk about the work that has been done towards solving these variations of the label placement problem. We also discuss the work done using other shapes of labels besides axis-parallel rectangles.

Kakoulis and Tollis have studied the case of labeling lines, or edges of a graph. They proved that the problem is NP-hard [KTo96]. They also studied labeling hierarchical graphs [KTo97], and the case in which more than one label is associated to an edge [KTo98-2].

Doddi et al. [DMM97] label rectilinear segments with rectangles of uniform width, and proved that the problem can be solved in $\Theta(n \log n)$ time when all segments are horizontal. Poon et al. [PZC98] gave an algorithm for labeling rectilinear segments of general orientation that takes $O(n \log n)$ time which matches the $\Omega(n \log n)$ lower bound of Doddi et al. [DMM97] and hence their algorithm is optimal.

Doddi et al. [DMM97] also provided approximation algorithms (see Chapter 7 for definition) using labels of different shapes such as circles and regular polygons, as well as using labels with different orientations. We discuss their results in more detail in the introduction of Chapter 7.

Kakoulis and Tollis [KTo98] and Wagner and Wolff [WWo98] presented general frameworks to label points, lines and areas, with labels of any shape and orientation.

Edmondson et al. [ECM97] presented some of the first results that combine the optimal placement of the labels with cartographic knowledge to set the candidate positions of the labels.

The goal of the study of all the different variants for the labeling problem is to combine them and be able to produce a good quality labeling for maps. In these maps we find cities (points), rivers (lines), regions (areas) and we must take care to follow the cartography rules to label each feature.

1.4 Overview

This thesis is concerned with labeling points on the plane using axis-parallel rectangles as labels. In Chapter 2, we present the definitions and notation that will be used throughout the thesis, as well as introduce an interesting variation of the point-feature label placement problem: the *elastic labeling problem*. This variation generalizes the point-feature label placement problem so that, in addition to choosing the corner of the label to place the point, we are free to choose the height and width as long as we achieve the specified area. This is of interest when the labels include information that consists of more than one word.

As well, in Chapter 2 we define the *sliding labels problem*, which generalizes the point-feature label placement problem so that any point on the boundary of the label, not just the corners, can be placed at the associated point. In this case each label has fixed dimension.

It is easy to see that the elastic labeling and the sliding labels problems are NP-hard since they generalize the point-feature label placement problem. Thus, to solve these problems we must either turn to heuristics and approximation methods, or add more constraints to the problem. In this thesis we consider both, but we focus on the latter approach.

The first part of this thesis focuses on practical variants of the elastic labeling

problem. In Chapters 3 and 4, we tackle two variations of the elastic labeling problem that are the main steps to solve the more general problem presented in Chapter 5.

In Chapter 3, we study the *two-axis labeling problem*. This is a special case of the elastic labeling problem in which the points lie on the positive x and y axes, and the labels lie in the first quadrant. We give a polynomial time dynamic programming algorithm for this problem.

In Chapter 4, we study the *two-parallel lines labeling problem*. This is a special case of the elastic labeling problem in which the points lie on two parallel lines and the labels lie between the lines. We also give a polynomial time algorithm for this problem.

In Chapter 5, we present the main result: a polynomial time algorithm for the elastic labeling problem when the given points lie on a boundary of a rectangular map. This *rectangle perimeter labeling problem* arises when the perimeter of the map is labeled with information about objects that lie beyond the boundary of the map, e.g. where the roads lead to. It is also applicable for labeling the coordinate axes in graphs such as charts and histograms. A practical application is, for example, GIS maps which are displayed dynamically on a computer screen, often using clipping, panning, and zooming, all of which benefit from perimeter labeling.

In Chapter 6, we present two sweep algorithms that solve some simple but nonetheless useful variations of the elastic labeling problem and the sliding labels problem.

In Chapter 7, we study approximation algorithms for the point-feature label placement problem.

In Chapter 8, we prove that the elastic labeling and the sliding labels problems

remain NP-hard even for the following constrained variants. For the elastic labeling problem we prove the NP-hardness of the *one-corner elastic labeling problem*, in which we fix the corner of each label that must be placed at the point-feature. For the sliding labels problem we prove the NP-hardness of the *left-right sliding labels problem*, even when only points on the left or right boundary of the label can be placed at the associated point. We also prove the NP-hardness of a variation of the point-feature label placement problem when there are at most three candidate positions to be considered per label.

Finally, we conclude the thesis in Chapter 9.

Chapter 2

Definitions and Notation

In this chapter we present the definitions and notation that will be used throughout the thesis. We define the *elastic labeling problem* and the *sliding labels problem*.

2.1 Elastic Labeling Problem

In this section, we propose an interesting alternative to the point-feature label placement problem, namely, the *elastic labeling problem* in which each label is a rectangle with fixed area, but varying height and width. First, we present a motivation for the use of elastic labels and then give definitions and notation that will be used throughout the thesis.

2.1.1 Motivation

An important and challenging task in cartography is the **labeling** of maps—attaching text to geographic features. Usually, the features to be labeled are regions, lines, and points on a map.

We can identify different kinds of labels. For example, Figure 2.1¹ depicts a portion of a map of Canada. In it there are point-feature (city) labels, e.g. **Calgary** and **Kelowna**, line-feature (river) labels, e.g. **Columbia**, region-feature labels, e.g. **Monashee Mountains**, and a special type of labels, *text labels*, giving information about the **Trans-Canada Highway** and **Glacier National Park**.



Figure 2.1: A Labeling Map.

Text labels are labels containing text consisting of several words. Figure 2.2²

¹Produced by the Cartographic Division of National Geographic Society.

²From: Mapping Hypertext. Robert E. Horn. The Lexington Institute, 1989.

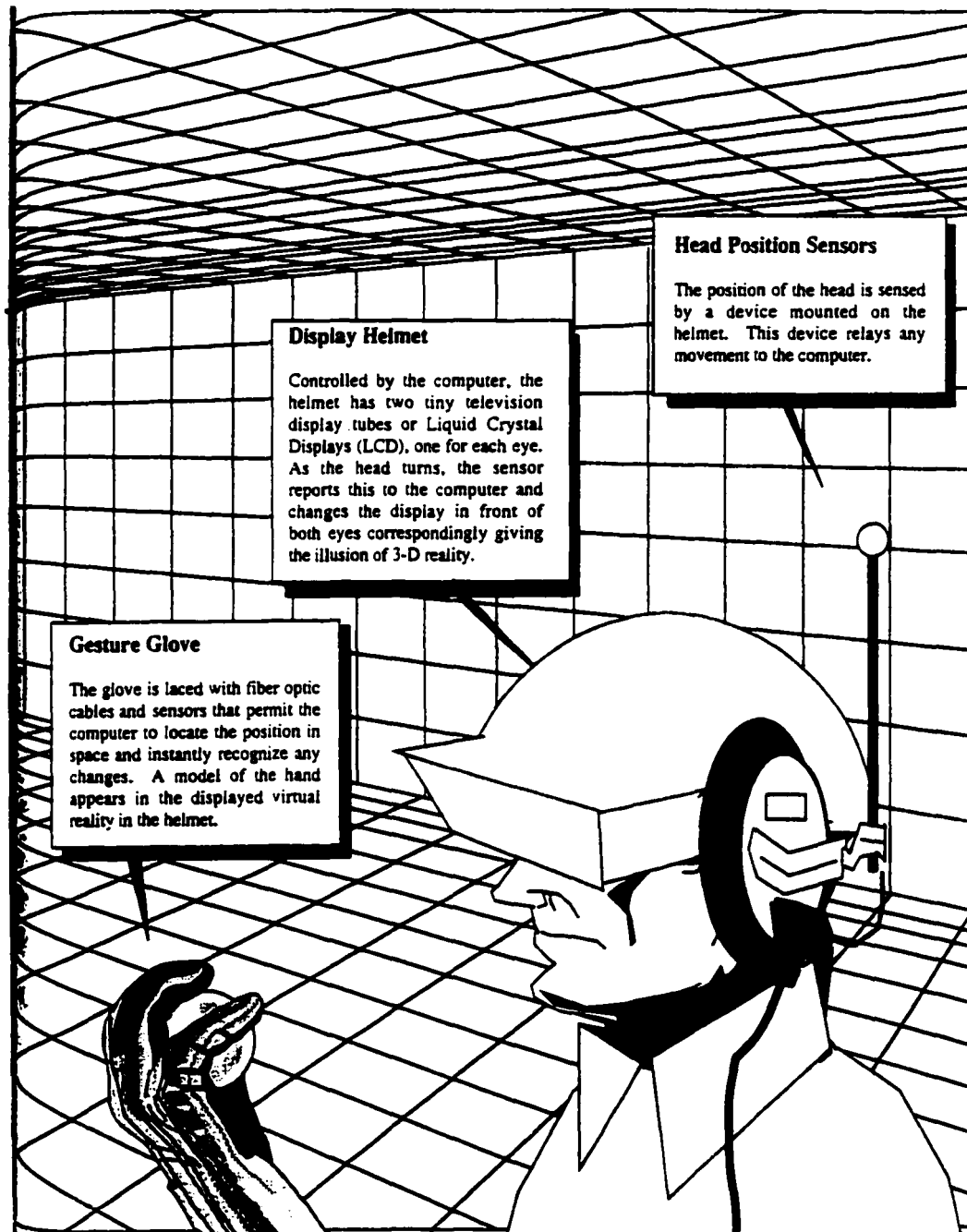


Figure 2.2: A diagram with text labels.

illustrates examples of text labels in a diagram. The relevant information in this figure is the information provided in the labels for the items presented: glove, helmet and sensors. In general, we could use any number of rows in each label so long as we have enough area for the text. In practice, we will have some restrictions on the labels to avoid very wide labels or very tall labels.

Text labels are very useful; we find text labels give information about touristic places, information about hospitals, descriptions of businesses, and so on.

With the idea of placing text labels on maps or diagrams we introduce a new type of labels, *elastic labels*, where each label is a rectangle with fixed area, but varying height and width. The objective is to use elastic labels to place text labels. Since text labels have more than one word we can write the specified text inside the label using one, two, or more rows, and then place the label at the specified point. For example a text label containing the legend “This is a label” can be written in one or two rows as shown in Figure 2.3. The labels shown in the figure have the same area.

In contrast with traditional labels, elastic labels offer more options for the label size of each object.



Figure 2.3: Using an elastic label to set a text label.

Now, we can proceed to introduce elastic labels into the label placement prob-

lem. We will use a traditional formulation of this problem, the point-feature label placement problem. Remember that for this problem, we are given a set of points in the plane and an axis-parallel rectangular label associated with each point. The problem consists of finding a placement for each label with one corner at its associated point such that no two labels overlap.

In this thesis, we propose an alternative to the problem by introducing “elastic labels”. Thus, in the *elastic labeling problem* we have a set of points each associated with an elastic label instead of a single rectangle. The problem consists of choosing the height and width of each label, and the corner placement at the associated point of each label, so that no two labels overlap.

The elastic labeling problem is NP-hard since it generalizes the point-feature label placement problem. That is, the problem is NP-hard even when there is no elasticity just because of the choice of the corners. The problem remains NP-hard even when we have elasticity but no choice about which corner of the label to use—we call this the *one-corner elastic labeling problem*. Thus, either elasticity or the choice of which corner of the label to use results in an NP-hard problem. In Chapter 8, we prove the NP-hardness of the one-corner elastic labeling problem.

Due to the NP-hardness result of the one-corner elastic labeling problem, if we want efficient algorithms we must either turn to heuristics and approximation methods (see Chapter 7), or add more constraints to the problem. In the first part of the thesis we take the latter approach, looking at a subproblem that is nonetheless useful. We require that the points be located on the boundary of a rectangle and their labels lie in the interior of the rectangle. Our main result is a polynomial time algorithm for this special case of the one-corner elastic labeling problem.

The *rectangle perimeter labeling problem*, presented in Chapter 5, arises when the perimeter of the map is labeled with information about objects that lie beyond the boundary of the map, e.g. where the roads lead to, etc. This problem is likely to be relevant in GIS as maps are displayed dynamically on a computer screen using clipping, panning, and zooming.

The London Underground Map³ shown in Figure 2.4, is a nice example of an application of the rectangle perimeter labeling problem. In this map we can see text labels located on the boundary of the rectangular map. These labels indicate where the routes beyond the map lead. For instance, on the bottom side of the map close to the right we can see that by following the green route we can reach Ealing, Broadway and Richmond; or if we follow the violet route indicated at the top of the map, we will reach Whitechapel and Upminster.

With the use of Geographical Information Systems (GIS) and Global Positioning Systems (GPS) we have maps that are clipped to fit their rectangular screens. Thus pointers that help us to navigate maps are needed. We can find tools for navigating these systems that allows movement on the map, but we need information about what is beyond the boundary of a map to know what place will be reached if we follow any direction beyond the boundary. Thus, we can see that our rectangle perimeter labeling problem has potential applications in the GIS and GPS markets.

Figure 2.5⁴ shows a map of a couple of blocks of Yorkville. This figure is also a good example for an application of our problem. Here the text labels give information about restaurants, shopping centers, and museums. These labels are placed on the boundaries of the blocks where the features are located.

³From: Let's Go Europe 1995. D. Fox, A. Dahlstein, and T. Tseng,
St. Martin's Press, New York.

⁴From: MapEasy, Inc. www.mapeasy.com.

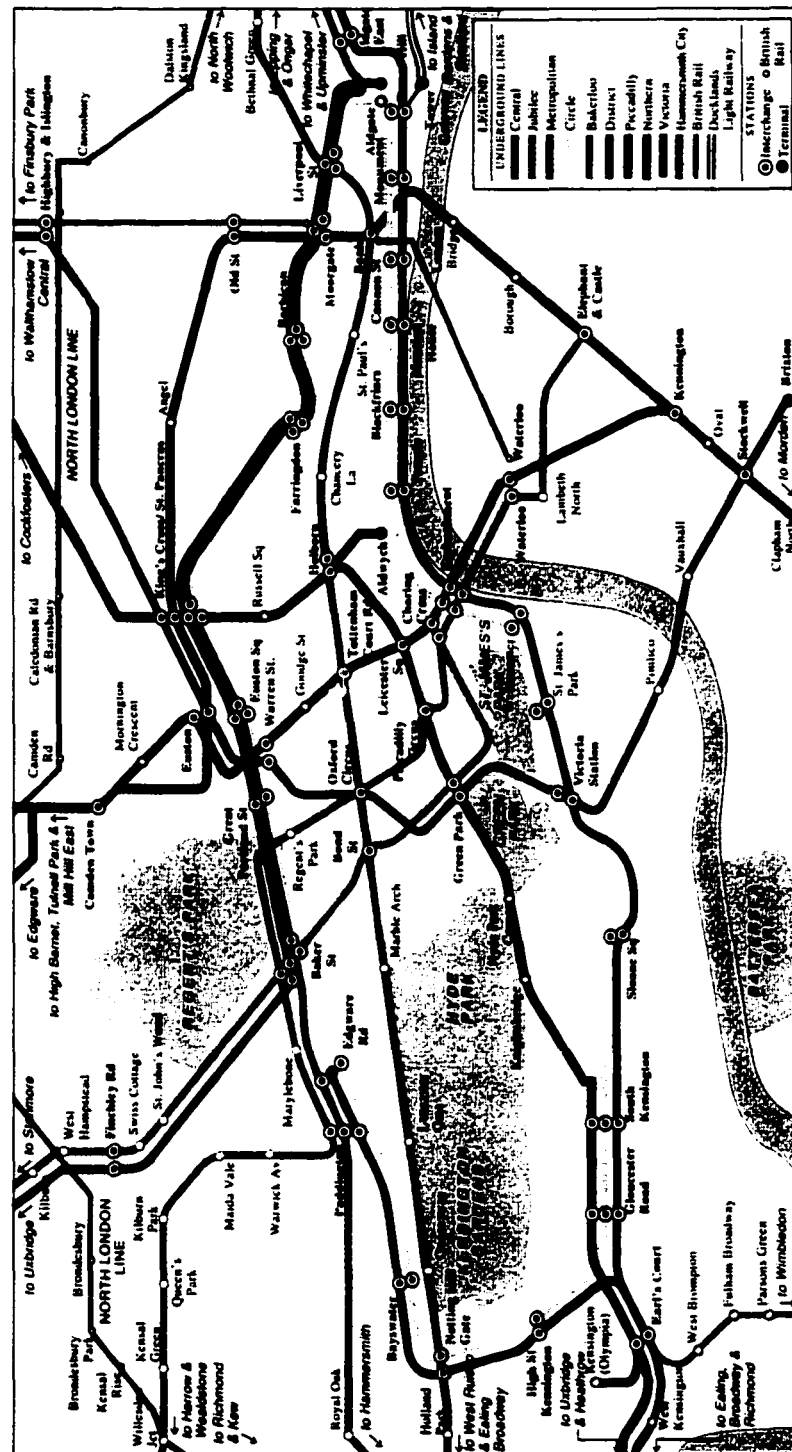


Figure 2.4: London Underground.

examples of this are automatic banking machines and navigational screens on helicopters, particularly on the U.S. Army's Comanche helicopter. The screen may display different layouts or maps selected by the user. The function of the buttons changes depending on what is displayed on the screen, so the screen must include a description of the function of each button. These descriptions must be placed around the perimeter of the screen, and can be modeled naturally as elastic labels.

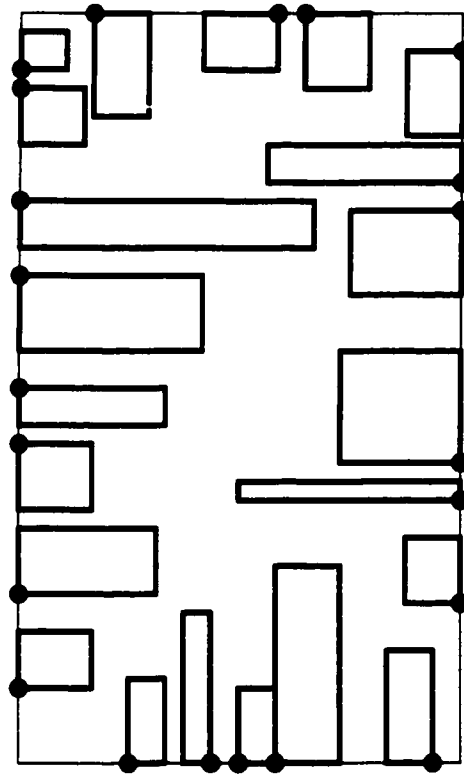


Figure 2.6: A solution for an instance of the rectangle perimeter labeling problem.

Now that we have discussed some applications of the rectangle perimeter labeling problem we show what a solution looks for an instance of the problem in Figure 2.6. To solve the rectangle perimeter labeling problem, we first tackle two subproblems in which the points lie on only two sides of the rectangle, either two

adjacent sides or two opposite sides. We call the case where the points lie on two adjacent sides of the rectangle the *two-axis labeling problem* (dealt with in Chapter 3), and the case where the points lie on two opposite sides of the rectangle the *two-parallel lines labeling problem* (dealt with in Chapter 4).

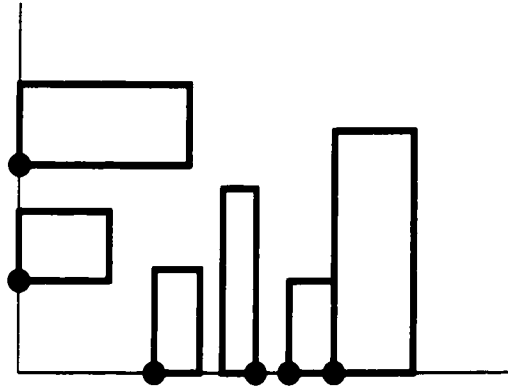


Figure 2.7: A solution for an instance of the two-axis labeling problem.

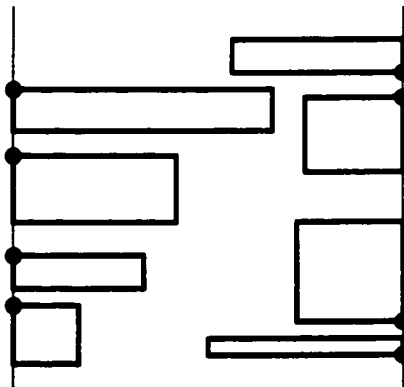


Figure 2.8: A solution for an instance of the two-parallel lines labeling problem.

For example, if we take contiguous portions of two adjacent sides, say bottom and left, of the boundary rectangle in Figure 2.6 we get Figure 2.7, and if we take facing portions of opposite sides, say left and right, we get Figure 2.8. These

two figures correspond to good realizations for instances of the two-axis labeling problem and the two-parallel lines labeling problem respectively.

Once we have algorithms for the two-axis labeling problem and the two-parallel lines labeling problem we need a method to combine them into a solution of the rectangle perimeter labeling problem. We will do this by showing that for any instance of the problem it suffices to consider a polynomial number of decompositions of the rectangle into regions in which labels from only two sides of the rectangle compete for space. (It is intuitively clear that near a corner of the map, for example, only the labels from two sides of the map are relevant.) Thus, we can apply our algorithms for the special cases in these regions. The dashed lines in Figure 2.9 illustrate one such decomposition.

In the following section we proceed to define more formally the concept of elastic labels.

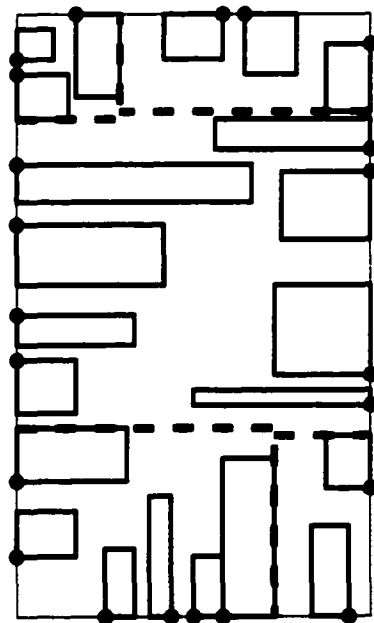


Figure 2.9: A decomposition.

2.1.2 Definitions and Notation

We define an orthogonal rectangle R by a quadruple (x, y, x', y') where the pair (x, y) sets the coordinates of the bottom left corner and the pair (x', y') sets the coordinates of the upper right corner. For a rectangle R we define $\text{left}(R) = x$, $\text{bottom}(R) = y$, $\text{right}(R) = x'$, $\text{top}(R) = y'$, $\text{height}(R) = y' - y$, and $\text{width}(R) = x' - x$. We consider rectangles to be topologically open.

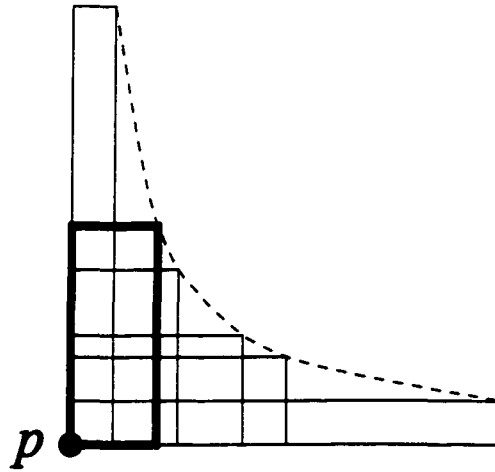


Figure 2.10: An elastic rectangle $\mathcal{E}$ with $Q(\mathcal{E}) = \{bl\}$.

Definition 2.1 An *elastic rectangle* $\mathcal{E}$ is a family of rectangles specified by a quintuplet (p, α, H, W, Q) where p is a point that is a corner of all rectangles in $\mathcal{E}$, α is the area of any rectangle in $\mathcal{E}$ (that is all rectangles in $\mathcal{E}$ have the same area), $H = [h^{\min}, h^{\max}]$ is the range of the height of the rectangles, $W = [w^{\min}, w^{\max}]$ is the range of the width, i.e. $\text{height}(E) \in [h^{\min}, h^{\max}]$ for all $E \in \mathcal{E}$ and $Q \subseteq \{bl, tl, tr, br\}$ is a set of possible positions of p allowed in the family. The value of the position is bl when p is a bottom left corner, tl when p is a top left corner, tr when p is a top right corner, and br when p is a bottom right corner.

We use the notation $p(\mathcal{E})$, $\alpha(\mathcal{E})$, $H(\mathcal{E})$, $W(\mathcal{E})$, and $Q(\mathcal{E})$ for the parameters of an elastic rectangle $\mathcal{E}$. The point $p(\mathcal{E})$ will be called the *anchor* of $\mathcal{E}$.

When Q is a singleton, the family of rectangles $\mathcal{E}$ is described by a hyperbola segment tracing out the locus of the corner of the elastic rectangle opposite p . Figure 2.10 shows an elastic rectangle with $Q(\mathcal{E}) = \{bl\}$, with the hyperbola as a dashed curve. Note that a discrete set of the rectangles in $\mathcal{E}$ are drawn but we are (for now) considering $\mathcal{E}$ to be infinite.

An elastic rectangle $\mathcal{E}$ with $Q(\mathcal{E}) = \{bl, tl, tr, br\}$ is illustrated in Figure 2.11. The dashed curves are the hyperbolas segments described by the opposite-to- p corners of the rectangles in the family.

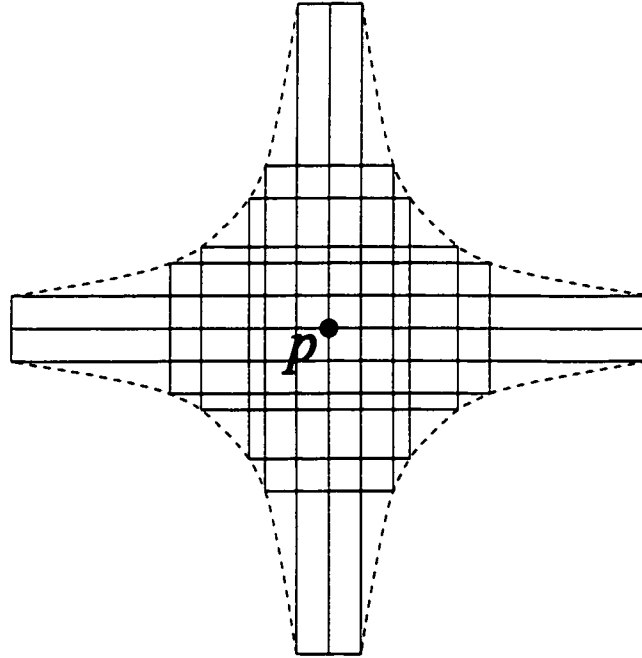


Figure 2.11: An elastic rectangle $\mathcal{E}$ with $Q(\mathcal{E}) = \{bl, tl, tr, br\}$.

We will freely apply the terms *tallest*, *widest*, and *thinnest* to elastic rectangles with their natural meaning.

Definition 2.2 A *realization* of an elastic rectangle $\mathcal{E}$, denoted E , is a single rectangle from the family—i.e. we must choose a valid height, width, and corner to place at $p(\mathcal{E})$.

We indicated in Figure 2.10 in bold a *realization* of the family.

We say that we *fix* an elastic rectangle when we choose one realization from its family. A realization of a set of elastic rectangles is called a *good realization* if the selected set of rectangles are pairwise disjoint (see Figure 2.14).

Definition 2.3 Given a set of elastic rectangles as input the *elastic labeling problem* is to find a good realization for the set.

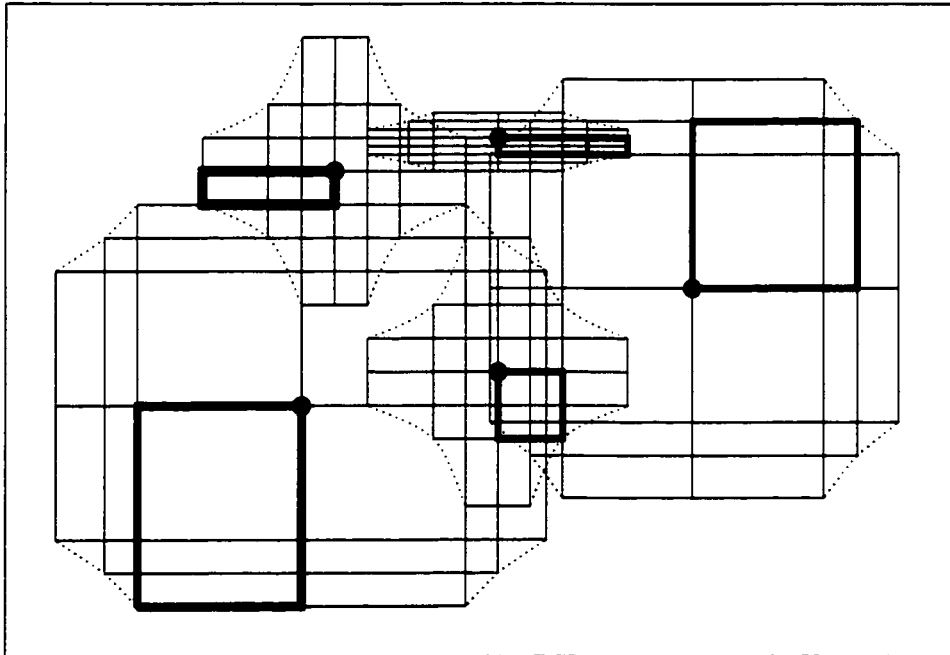


Figure 2.12: An instance of the elastic labeling problem.

An instance for the elastic labeling problem is shown in Figure 2.12, where the rectangles indicated in bold form a good realization. Notice that in general there

might be more than one good realization.

To gain more intuition about the use of elastic labels we take a bad realization of the instance in Figure 2.12, shown in Figure 2.13. In this figure the labels assigned to points p and q intersect. To obtain a good realization, we can either make use of the elasticity as shown in Figure 2.14, by stretching the width of the rectangle at q ; or changing the corner selection of p , to be a top-right corner instead of a bottom-left corner of its label, as shown in Figure 2.15.

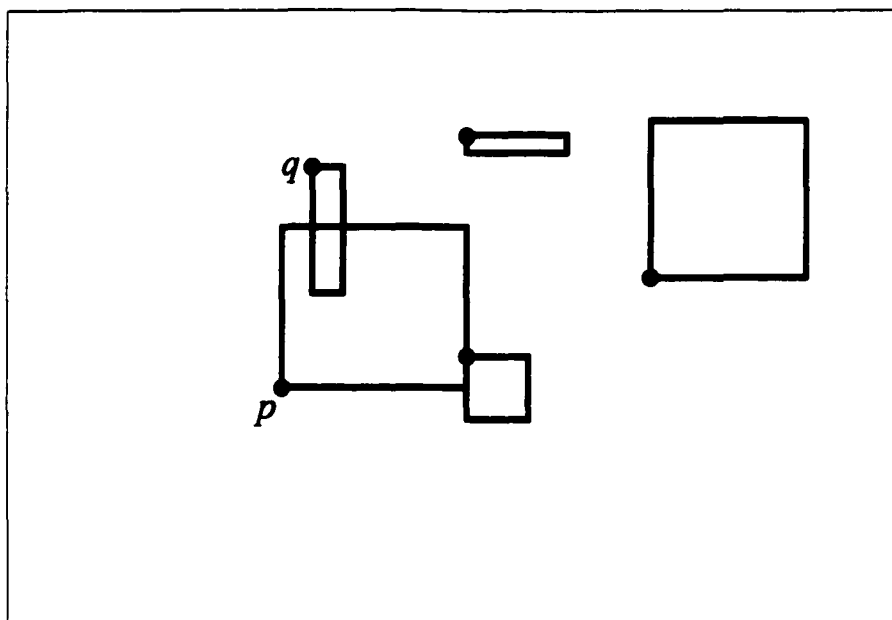


Figure 2.13: A bad realization for the instance of Figure 2.12.

Definition 2.4 A *one-corner elastic rectangle* is an elastic rectangle with $|Q| = 1$.

Definition 2.5 The *one-corner elastic labeling problem* is a special case of the elastic labeling problem where all the elastic rectangles are one-corner elastic rectangles.

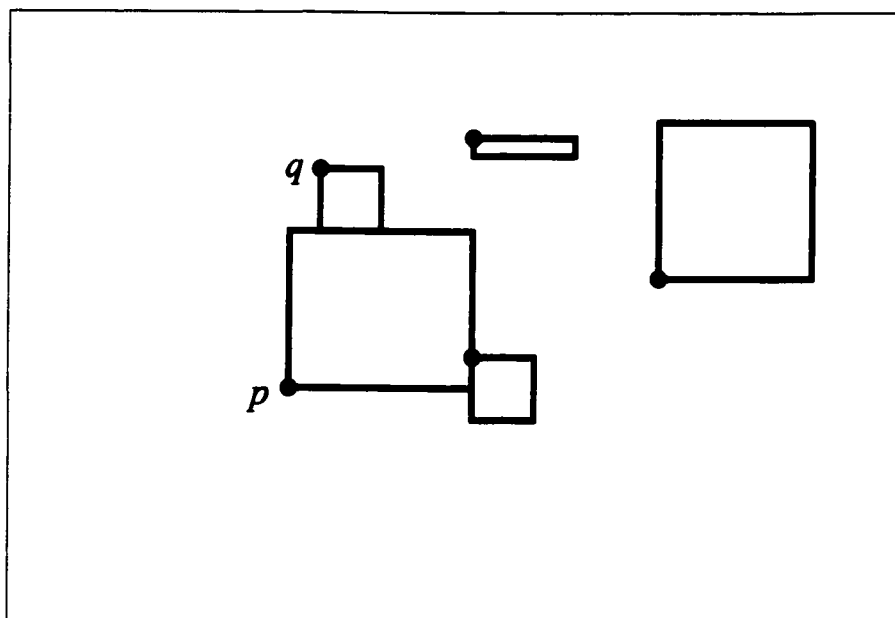


Figure 2.14: Good realization obtained by using elasticity.

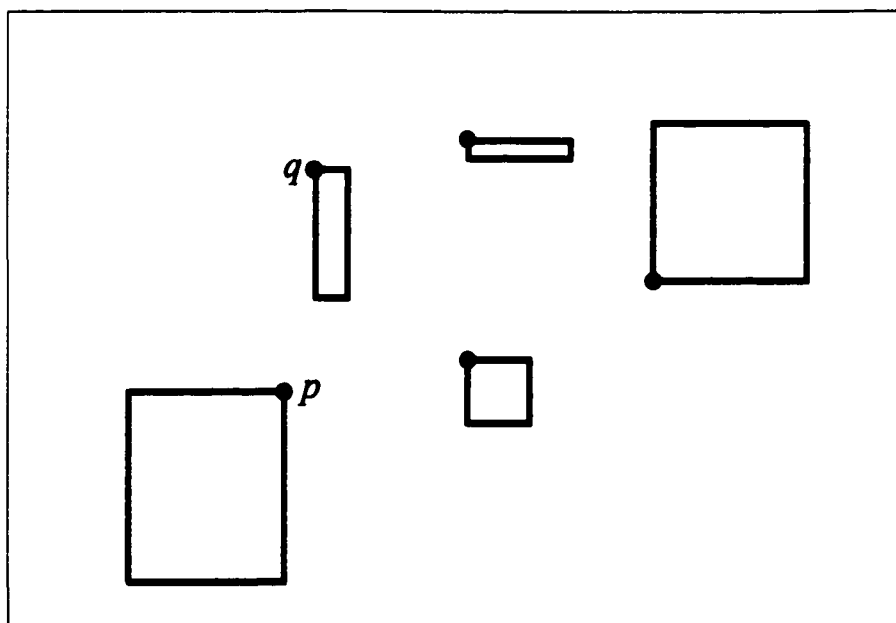


Figure 2.15: Good realization obtained by changing corner selection.

2.2 Sliding Labels Problem

In this section we define the *sliding labels problem*, which generalizes the point-feature label placement problem so that any point of the boundary of the label, not just the corners, can be placed at the associated point.

2.2.1 Motivation and Definitions

The sliding labels problem is a traditional labeling problem in the sense that the labels are not elastics (although not necessarily all of the same size) and we have to position them so that they do not overlap. The concept of sliding labels was introduced by Hirsch [Hir82]. In contrast with an elastic label, a sliding label has a fixed width and height. In the *sliding labels problem* any point of the boundary of a label, not just the corner, can be placed at its associated point. We can see some examples of sliding labels in Figure 2.16.

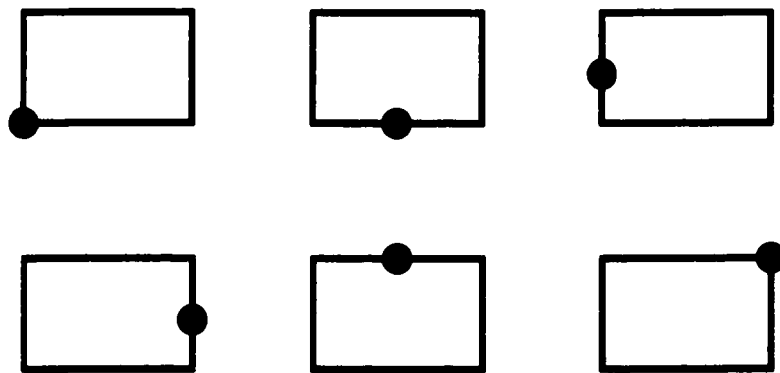


Figure 2.16: Sliding Labels.

More formally we define the sliding labels problem as follows:

Definition 2.6 *Given a set of points in the plane, each point associated with an*

axis-parallel rectangle, the **sliding labels problem** is to place the rectangles such that their associated points are on their boundary and the rectangles are pairwise disjoint.

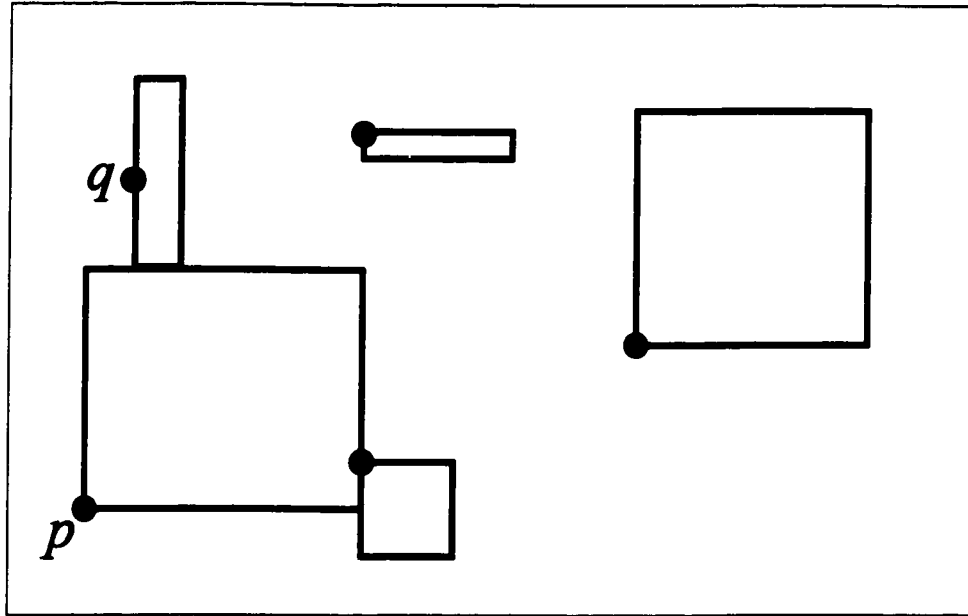


Figure 2.17: A good realization for an instance of the sliding labels problem.

Figure 2.17 depicts a good realization of Figure 2.13 that we obtain by sliding the label of point q up.

Van Kreveld et al. compared the sliding labels problem with the traditional point-feature label placement problem [KSW98]. They also provided a 2-approximation algorithm (see Chapter 7 for definition) for rectangles of the same height. Also for this problem Hirsch [Hir82] gave a heuristic that uses repelling forces for overlapping rectangles and computes translations vectors for them. (The translation process is repeated until a labeling with few overlaps appears.)

It is easy to see that the sliding labels problem is NP-hard since it generalizes the

point-feature label placement problem. In Chapter 8, we show that this problem remains NP-hard even when only points on the left or right boundary of the label can be placed at the associated point: the *left-right sliding labels problem*. Adding to this problem the constraint that all the labels must be placed at the same side makes the problem solvable. This variant of the problem is still a practical one. We present, in Chapter 6, a sweep algorithm that solves it.

We will use notation similar to that of section 2.1.2. Rectangles are considered to be topologically open. The point associated with a rectangle is called its *anchor*. A *good realization* for a set of rectangles is a placement of the rectangles at their anchors such that the rectangles are pairwise disjoint.

Chapter 3

The Two-Axis Labeling Problem

In this chapter we give a polynomial time algorithm for a special case of the one-corner elastic labeling problem, where the anchors of the elastic rectangles lie on the positive x and y axes and the labels lie in the first quadrant.

3.1 Definitions and Preliminaries

An elastic rectangle $\mathcal{E}$ is x -base if the bottom sides of its rectangles lie on the positive x -axis and $Q(\mathcal{E}) = \{bl\}$ or $Q(\mathcal{E}) = \{br\}$. Similarly $\mathcal{E}$ is y -base if the left sides of its rectangles lie on the positive y -axis and $Q(\mathcal{E}) = \{bl\}$ or $Q(\mathcal{E}) = \{tl\}$.

Definition 3.1 *Given a set of x -base elastic rectangles, and a set of y -base elastic rectangles the **two-axis labeling problem** is to find a good realization for the union of the two sets (see Figure 3.1).*

We indicate that the elastic rectangles that have been fixed in our figures by using bold rectangles.

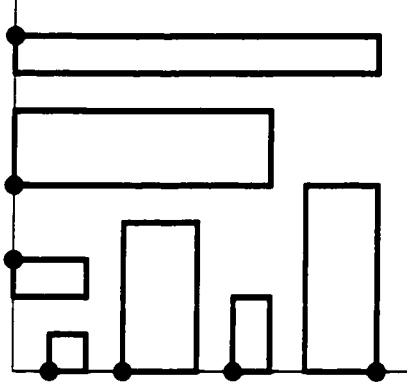


Figure 3.1: A good realization for the two-axis labeling case.

Given a point in the plane p_i with coordinates (x_i, y_i) , we define $x(p_i) = x_i$ and $y(p_i) = y_i$.

Let $B = \{B_1, \dots, B_n\}$ be a set of x -base elastic rectangles with $p_i = p(B_i)$ and $x(p_i) \leq x(p_{i+1})$ for $i = 1, \dots, n-1$. Let $L = \{L_1, \dots, L_m\}$ be a set of y -base elastic rectangles with $q_i = p(L_i)$ and $y(q_i) \leq y(q_{i+1})$ for $i = 1, \dots, m-1$.

We define the *height* [*width*] of a realization as the height [*width*] of the smallest axis-parallel rectangle containing all the rectangles of the realization.

The algorithm for the two-axis labeling problem adds points one by one. To gain some intuition about where the difficulty of the problem lies, observe that when two consecutive points along the x -axis have their rectangles facing each other (as in Figure 3.2), we can trade-off the height of one against the height of the other, and which solution is best depends on the particular y -base rectangles we have.

We propose a dynamic programming algorithm for the two-axis labeling problem in which we add points one by one along the x and y axes. We will begin in Section 3.2 by describing the global structure of a solution. This allows us to identify the subproblems to be solved in the dynamic programming algorithm.

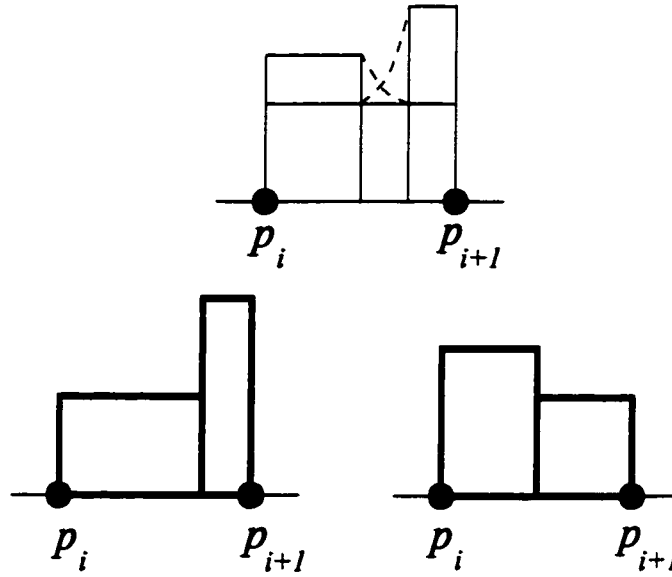


Figure 3.2: Elastic rectangles facing each other.

Section 3.3 contains the complete algorithm.

In our figures we indicate the range of an elastic rectangle by a hyperbola segment located in the opposite corner of its anchor.

3.2 Structure of a Good Realization

The dynamic programming algorithm hinges on the following.

Observation 3.1 *In a good realization of $\mathcal{B}_1, \dots, \mathcal{B}_n$ and $\mathcal{L}_1, \dots, \mathcal{L}_m$, using rectangles $B_1, \dots, B_n$ and $L_1, \dots, L_m$, respectively, either B_n is top-clear—i.e. all the other rectangles lie to the left of the x -coordinate $\text{left}(B_n)$ —or L_m is right-clear—i.e. all the other rectangles lie below the y -coordinate $\text{bottom}(L_m)$. See Figure 3.3.*

We call a realization *top-clear* if its last x -base rectangle is top-clear; we call a realization *right-clear* if its last y -base rectangle is right-clear. Note that a realization

can be both top-clear and right-clear.

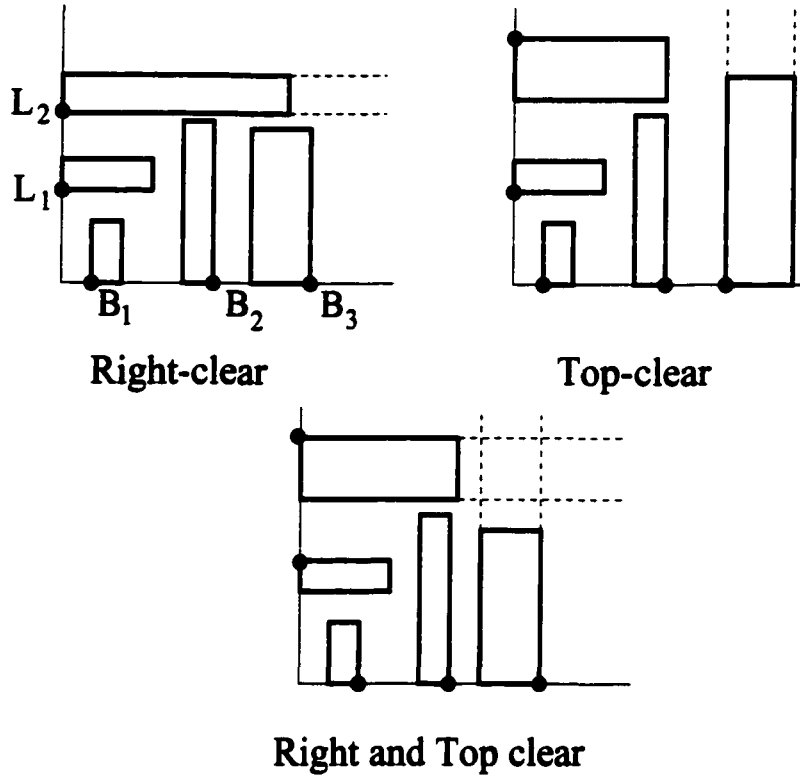


Figure 3.3: Good clear realizations.

The above observation tells us that we can always “peel away” the rectangles of a good realization, removing a top-clear or right-clear one. For example in Figure 3.4a we have a top-clear realization. By peeling away the top-clear rectangle we obtain the good realization shown in Figure 3.4b, which is both top and right clear. We can then peel away either of the two clear rectangles, if we peel away the top-clear rectangle we obtain the realization of Figure 3.4c, which is a right-clear realization.

This peeling away process introduces an ordering (actually, its reverse) that will drive our dynamic programming algorithm. More formally, let $S_{i,j}$ be the subproblem of finding a good realization for $B_1, \dots, B_i, L_1, \dots, L_j$. Let $S_{0,0}$ be the

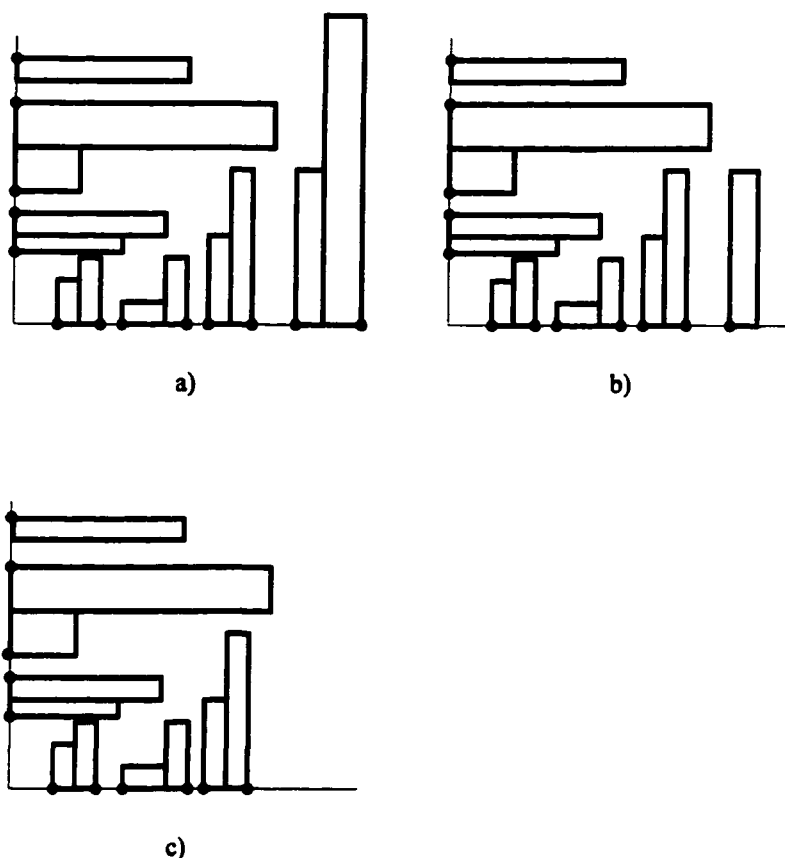


Figure 3.4: Peeling way rectangles in a clear realization.

null subproblem.

Theorem 3.1 Suppose $B_1, \dots, B_n, L_1, \dots, L_m$ is a good realization of an instance of the two-axis labeling problem, and let $S_{i,j}$ be the good realization induced for the subproblem $S_{i,j}$. Then there is a sequence of pairs of indices $(0, 0), \dots, (n, m)$ such that, if (i, j) appears in the sequence then the next element of the sequence is either $(i + 1, j)$ and the good realization $S_{i+1,j}$ is top-clear, or $(i, j + 1)$ and the good realization $S_{i,j+1}$ is right-clear.

The proof of this theorem follows from Observation 3.1. We call such a sequence of pair of indices a *clear decomposition*. For example, in Figure 3.5a a clear decom-

position is $(0,0)$, $(1,0)$, $(2,0)$, $(2,1)$, $(3,1)$, $(3,2)$, $(4,2)$, $(4,3)$, $(5,3)$, $(6,3)$, $(6,4)$, $(7,4)$, $(7,5)$, $(8,5)$. In Figure 3.5b we have in dotted bold lines good realizations defined for each pair of indices in the clear decomposition. Note that in general, the clear decomposition is not unique, we can replace $(6,4)$ by $(7,3)$ in this sequence and obtain another clear decomposition of the same realization.

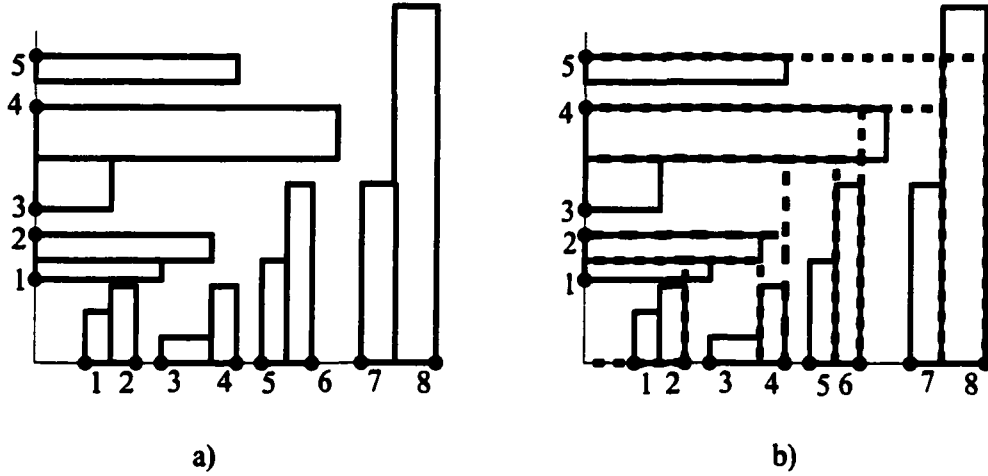


Figure 3.5: Clear decomposition of a good realization.

3.3 Algorithm for the Two-Axis Labeling Problem

The subproblems solved by the dynamic programming algorithm are as follows. For each $i, j, 0 \leq i \leq n, 0 \leq j \leq m$ we will solve two subproblems $\mathcal{S}_{i,j}^T$ and $\mathcal{S}_{i,j}^R$. For $\mathcal{S}_{i,j}^T$ we will find the “best” good realizations of $\mathcal{B}_1, \dots, \mathcal{B}_i, \mathcal{L}_1, \dots, \mathcal{L}_j$ with $\mathcal{B}_i$ top-clear. For $\mathcal{S}_{i,j}^R$ we will find all the “best” good realizations for $\mathcal{B}_1, \dots, \mathcal{B}_i, \mathcal{L}_1, \dots, \mathcal{L}_j$ with $\mathcal{L}_j$ right-clear. (Note that for $i, j \geq 1$, $\mathcal{S}_{0,j}^T$ and $\mathcal{S}_{i,0}^R$ have no solutions; however, we do consider the null set to be a solution to $\mathcal{S}_{0,0}^T$ and $\mathcal{S}_{0,0}^R$). What do we mean

by “best”? Since we will only ever add top-clear x -base rectangles and right-clear y -base rectangles, the only things that matter about a realization are its height and width. Recall that the height [width] of a realization is the maximum height [width] of a rectangle in the realization. The height of the realization matters for adding a right-clear rectangle, and the width matters for adding a top-clear rectangle.

In case there is a unique minimum height and width for a good realization, we want a good realization of that height and width. The good realization itself is in general not unique, but that does not matter. The left pane of Figure 3.7 shows a case where there is a unique minimum height and width top-clear good realization.

Often there is no unique minimum height and width—height and width may be traded off as they are in a single elastic rectangle. For example Figure 3.6 depicts this situation. In this figure we have two elastic rectangles $\mathcal{B}_1$ and $\mathcal{L}_1$ and, as is shown in the right pane of the figure with bold dashed lines, if we take the tallest rectangle in $\mathcal{B}_1$ we have a top-clear good realization with minimum width, but if we take the widest rectangle then we have a top-clear good realization with minimum height. In this case we preserve all possible height-width combinations. As we shall see, we can fix the realizations of all rectangles except the last one (the clear one), which must remain elastic.

To be more specific, in solving $\mathcal{S}_{i+1,j}^T$ we are adding $\mathcal{B}_{i+1}$ as a top-clear rectangle. If $\mathcal{B}_{i+1}$ ’s anchor is to its right, we will find a good realization of $\mathcal{B}_1, \dots, \mathcal{B}_{i+1}, \mathcal{L}_1, \dots, \mathcal{L}_j$ of minimum height and width, with $\mathcal{B}_{i+1}$ top-clear. To justify this, note that since the width of any good realization is fixed (namely at p_{i+1}), we can simply minimize the height, and fix a good realization. On the other hand, if $\mathcal{B}_{i+1}$ ’s anchor is to its left then we will find one good realization of $\mathcal{B}_1, \dots, \mathcal{B}_i, \mathcal{L}_1, \dots, \mathcal{L}_j$ of width no more than p_{i+1} of minimum height, together with a restricted range for the elastic rectangle $\mathcal{B}_{i+1}$. It makes sense to leave $\mathcal{B}_{i+1}$ elastic, since as noted

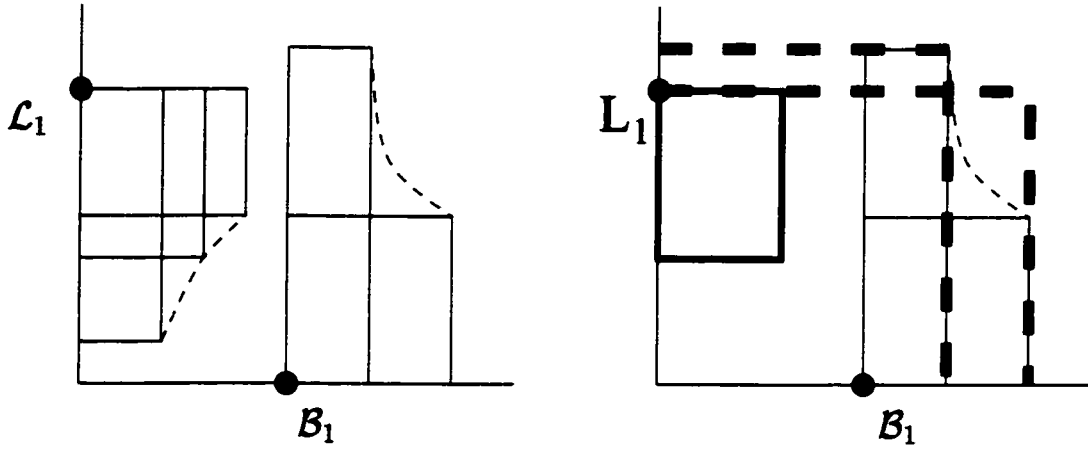


Figure 3.6: A good realization with a non-unique minimum height and width.

before, the realization we choose for $\mathcal{B}_{i+1}$ determines the realizations we can choose for the next x -base or y -base rectangle that we add. On the other hand, it makes sense to fix realizations of all prior rectangles, since we now have a bound on the width, and can simply minimize height.

Using the arguments discussed above we proceed to explain our algorithm in the next section.

3.4 Two-axis algorithm.

Assume we have solved subproblems $\mathcal{S}_{i,j}^T$ and $\mathcal{S}_{i,j}^R$. This is how we solve subproblem $\mathcal{S}_{i+1,j}^T$. (We use a symmetric solution for $\mathcal{S}_{i,j+1}^R$.)

We explain the case when we add $\mathcal{B}_{i+1}$ to form a top-clear realization. There are two cases to be considered: either p_{i+1} is on the left of the elastic rectangle or on its right. In the first case the new rectangle will remain elastic in our solution; in the second case, because the width of the solution is fixed, we can simply minimize the height, and will fix $\mathcal{B}_{i+1}$. See Figure 3.7.

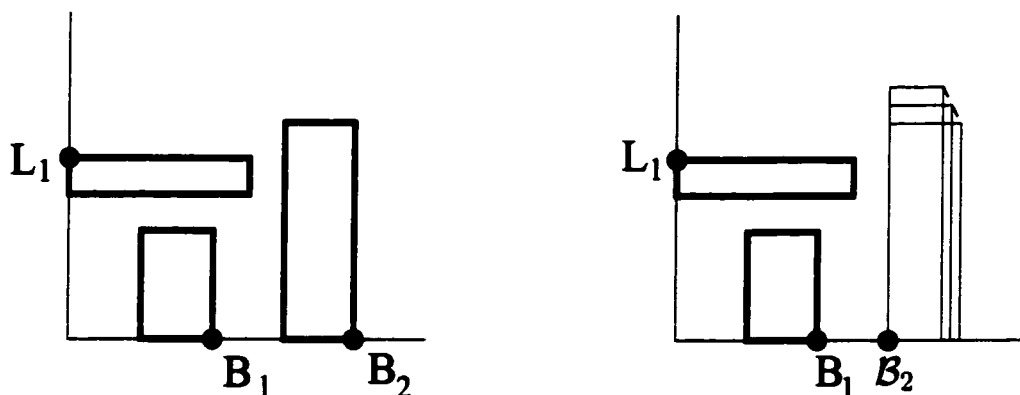


Figure 3.7: Adding elastic rectangle B_2 to a right-clear realization.

Case I. Suppose $p = p_{i+1}$ is at the left of B_{i+1} . We may build on $\mathcal{S}_{i,j}^T$ or on $\mathcal{S}_{i,j}^R$. We compare them to see which is best. We need a solution to $\mathcal{S}_{i,j}$ with width no more than $x(p)$ (see Section 3.1 for definition of $x(p)$). From the stored solutions to each of $\mathcal{S}_{i,j}^T$ and $\mathcal{S}_{i,j}^R$, take a good realization with width less than or equal to $x(p)$ and with minimum height. Of these two solutions, keep the one of minimum height h . We now add B_{i+1} , keeping it as an elastic rectangle, though with a height and width range that may be restricted. Since the height of any good realization must be at least h , we want the minimum width solution of height h , and all the possible trade-offs for solutions of height greater than h . Thus, if the maximum height of B_{i+1} is less than h , we fix B_{i+1} to the maximum possible height, and otherwise, we restrict the height range of B_{i+1} from h on up and take the corresponding width range.

Case II. Suppose p_{i+1} is at the right of B_{i+1} . The width of the solution is then fixed (namely at p_{i+1}) and we are looking for the solution of minimum height. We will fix B_{i+1} . We may build on $\mathcal{S}_{i,j}^T$ or on $\mathcal{S}_{i,j}^R$. We compare them and see which of them provide us with the best—i.e. minimum height—solution.

Case II.1. Solving $\mathcal{S}_{i+1,j}^T$ building on the solution for $\mathcal{S}_{i,j}^T$. If we have a unique

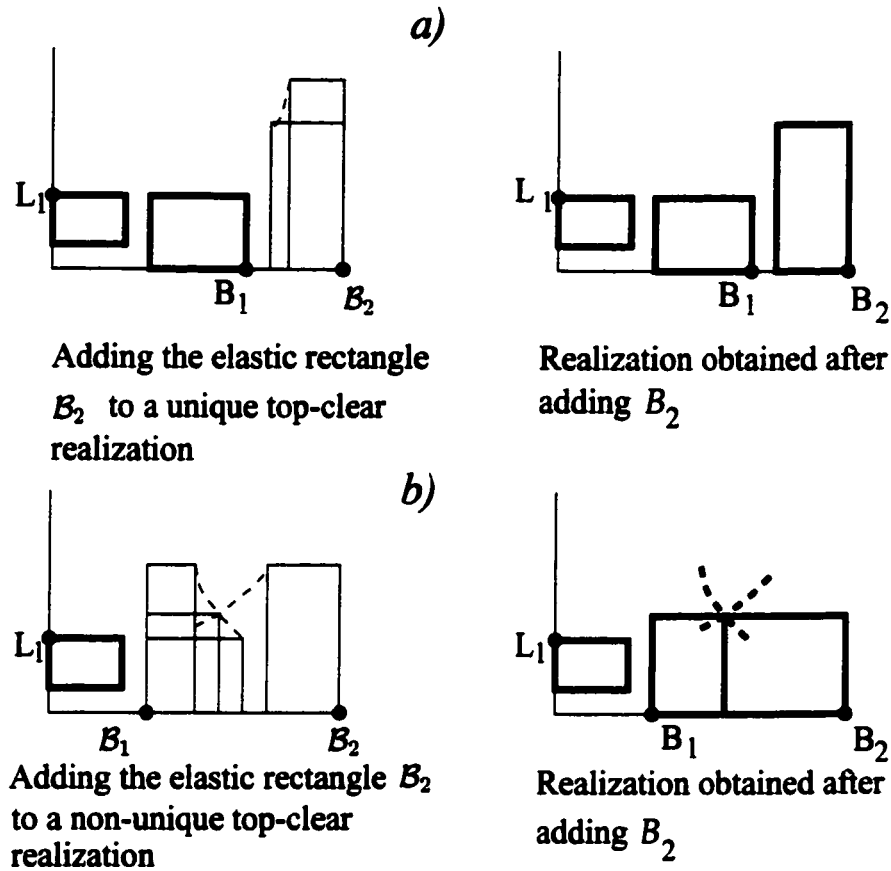


Figure 3.8: Case I.1.

realization stored for this subproblem, then we simply add the minimum height realization for B_{i+1} that fits, i.e. that whose left side is greater than or equal to the width of the subproblem's realization (Figure 3.8a). Otherwise, B_i is still elastic in the solution to the subproblem. In this case we must find the minimum height good realization for two x -base elastic rectangles B_i and B_{i+1} with their anchors on the left and right respectively. See Figure 3.8b. This involves computing the intersection of two hyperbolas as bounded by range constraints. The details of the computation can be found in the following section.

Case II.2. Solving $\mathcal{S}_{i+1,j}^T$ building on the solution for $\mathcal{S}_{i,j}^R$. If we have a unique

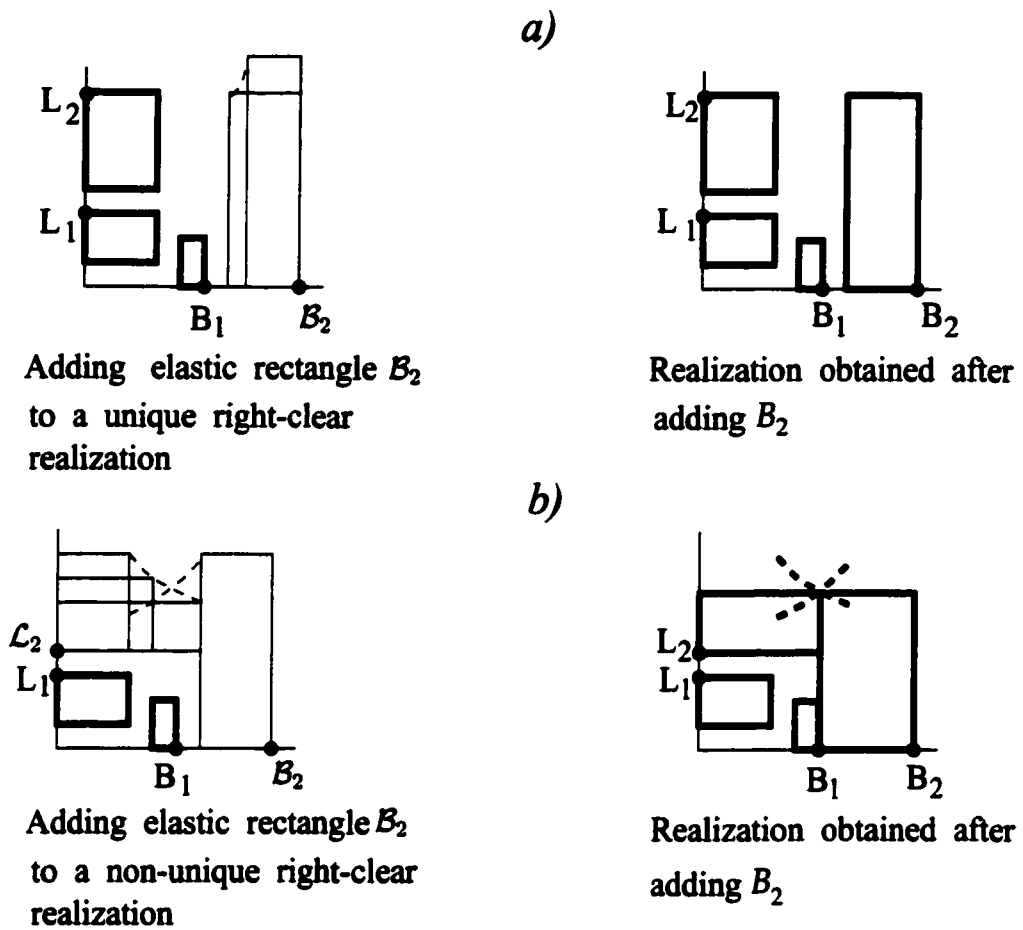


Figure 3.9: Case I.2.

realization stored for this subproblem, then we simply add the minimum height realization that fits, i.e. whose left side is greater than or equal to the width of the subproblem's realization (Figure 3.9a). Otherwise, $\mathcal{L}_j$ is still elastic in the solution to the subproblem. In this case we must find the minimum height good realization for two elastic rectangles $\mathcal{L}_j$ and B_{i+1} with their anchors at the bottom-left and bottom-right respectively. See Figure 3.9b. This is very similar to the problem we solved in Case II.1—the only difference is that the first elastic rectangle does not lie on the x -axis. The details of the computation can be found in section 3.5.

This completes the description of the main step of the algorithm. We start the algorithm off with the solutions to the subproblems $\mathcal{S}_{0,0}^T$ and $\mathcal{S}_{0,0}^R$ as rectangles of 0 width and height at the origin. When the algorithm has solved $\mathcal{S}_{n,m}^T$ and $\mathcal{S}_{n,m}^R$ we learn whether there is a solution to the whole problem, or not.

It is easy to see that the algorithm takes $O(nm)$ time.

We use the following lemma to prove the correctness of the algorithm.

Lemma 3.1 *The solution to subproblem $\mathcal{S}_{i,j}^T$ captures all minimal height-width combinations for good realizations of $\mathcal{B}_1, \dots, \mathcal{B}_i, \mathcal{L}_1, \dots, \mathcal{L}_j$ with $\mathcal{B}_i$ top-clear. As well, the solution to subproblem $\mathcal{S}_{i,j}^R$ captures all minimal height-width combinations for good realizations of $\mathcal{B}_1, \dots, \mathcal{B}_i, \mathcal{L}_1, \dots, \mathcal{L}_j$ with $\mathcal{L}_j$ right-clear. (By minimal, we mean minimal in the ordering $(h, w) \leq (h', w')$ if $h \leq h'$ and $w \leq w'$.)*

PROOF. We prove both statements simultaneously by induction on $i + j$, using as our basis the case $i + j = 0$, for which the null set is a solution to $\mathcal{S}_{0,0}^T$ and $\mathcal{S}_{0,0}^R$.

Assume the statements hold for $\mathcal{S}_{i,j}^T$ and $\mathcal{S}_{i,j}^R$, for all $i + j = k$. Consider $\mathcal{S}_{i+1,j}^T$, $i + j = k$. (The case for $\mathcal{S}_{i,j+1}^R$ is symmetric.) We are adding $\mathcal{B}_{i+1}$, and we follow the cases of the algorithm above.

Case I. $p = p_{i+1}$ is at the left of $\mathcal{B}_{i+1}$. Then we need a solution for subproblem $\mathcal{S}_{i,j}$ with width smaller or equal to $x(p)$. By induction hypothesis we know that solutions $\mathcal{S}_{i,j}^T$ and $\mathcal{S}_{i,j}^R$ capture all possible minimum height-width combinations for top-clear and right-clear realizations respectively for subproblem $\mathcal{S}_{i,j}$. Thus, if there is a good realization for subproblem $\mathcal{S}_{i,j}$ with width smaller or equal to $x(p)$ solutions $\mathcal{S}_{i,j}^T$ and $\mathcal{S}_{i,j}^R$ capture it and with the minimum possible height. From Observation 3.1 for solution $\mathcal{S}_{i+1,j}^T$ we only need to consider $\mathcal{S}_{i,j}^T$ and $\mathcal{S}_{i,j}^R$. Therefore, the height of $\mathcal{S}_{i+1,j}^T$ is at least the height of the minimum height realization from

$S_{i,j}^T$ or $S_{i,j}^R$ with width smaller than or equal to $x(p)$. Let S be this realization with height h .

Then, by placing S together with $\mathcal{B}_{i+1}$ we obtain all possible minimal height-width combinations by fixing $\mathcal{B}_{i+1}$ to its realization of maximum height if its height is smaller than h , or otherwise by considering the realizations of $\mathcal{B}_{i+1}$ with height greater than or equal to h .

Case II. $p = p_{i+1}$ is at the right of $\mathcal{B}_{i+1}$. Then the width of $S_{i+1,j}^T$ is fixed by $x(p)$. Thus, we only have to look for a top-clear realization of minimum height between the ones obtained from either $S_{i,j}^T$ adding $\mathcal{B}_{i+1}$ or $S_{i,j}^R$ adding $\mathcal{B}_{i+1}$. We explain the case where we obtain a realization from $S_{i,j}^T$ adding $\mathcal{B}_{i+1}$. The other case is similar. We have that either $S_{i,j}^T$ has one rectangle remaining elastic, $\mathcal{B}_i$, or does not. If $\mathcal{B}_i$ remains elastic then we obtain a good realization of minimum total height for both $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$. If $\mathcal{B}_i$ does not remain elastic then we can think of $S_{i,j}^T$ as a single fixed rectangle and obtain the realization of minimum total height for it together with $\mathcal{B}_{i+1}$. By the induction hypothesis $S_{i,j}^T$ captures all minimal height-width combinations for a top-clear realization of elastic rectangles $\mathcal{B}_1, \dots, \mathcal{B}_i$ and $\mathcal{L}_1, \dots, \mathcal{L}_j$. Thus, we have obtained a top-clear realization that captures all minimal height-width combinations for $\mathcal{B}_1, \dots, \mathcal{B}_{i+1}, \mathcal{L}_1, \dots, \mathcal{L}_j$ that contains realization $S_{i,j}^T$.

The above arguments are applied similarly to the realization obtained for $S_{i,j}^R$ adding $\mathcal{B}_{i+1}$.

From Observation 3.1 a top-clear realization $S_{i+1,j}^T$ is built from $S_{i,j}^T$ adding $\mathcal{B}_{i+1}$ or $S_{i,j}^R$ adding $\mathcal{B}_{i+1}$. By the induction hypothesis we have all possible minimal height-width combinations for $S_{i,j}^T$ and $S_{i,j}^R$. Therefore, by selecting the top-clear realization of minimum height between the ones obtained from either $S_{i,j}^T$ adding $\mathcal{B}_{i+1}$ or $S_{i,j}^R$ adding $\mathcal{B}_{i+1}$, we get a realization $S_{i+1,j}^T$ of minimum height. $\square$

The correctness of the algorithm follows from Theorem 3.1 which guarantees a clear decomposition, and justifies solving subproblem $\mathcal{S}_{i+1,j}^T$ by only looking at solutions to $\mathcal{S}_{i,j}^T$ and $\mathcal{S}_{i,j}^R$; and by Lemma 3.1, which tells us about the form of the solution to subproblem $\mathcal{S}_{i+1,j}^T$. Therefore, the following theorem holds.

Theorem 3.2 *Given an instance of the two-axis labeling problem the two-axis algorithm finds a good realization if and only if one exists in $O(nm)$ time, where n and m are the numbers of x -base and y -base rectangles respectively.*

3.5 Hyperbolas

In this section we will explain the one missing detail of the algorithm, computing the intersection of two hyperbolas. We use this intersection to compute a minimum height good realization for two elastic rectangles. We use a real-RAM model of computation.

First, we explain the case when the two elastic rectangles are located in the same axis and then, when they are in opposite axes.

3.5.1 Elastic Rectangles in the Same Axis

Assume that we have two x -base elastic rectangles, $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$, with their anchors on the left and right respectively. We wish to find realizations of the two elastic rectangles with minimum total height—i.e. minimizing the maximum height of the two. This problem arises in case II, subcase 1 of the algorithm (see Figure 3.8b). We call $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$ *facing x -base rectangles*.

Given a good realization of a pair of facing elastic rectangles on the same axis, we define its *total height* to be the maximum of the heights of the two rectangles.

Let H be the total height of realizations of $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$, which we want to minimize. Let a and b be the areas of $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$ respectively. Let h_i and h_{i+1} be the heights, and w_i and w_{i+1} be the widths of realizations $B_i \in \mathcal{B}_i$ and $B_{i+1} \in \mathcal{B}_{i+1}$ respectively. Then our problem is described as follows:

$$\min H$$

Subject to:

$$h_i \leq H$$

$$h_{i+1} \leq H$$

$$h_i = \frac{a}{x_i - p(\mathcal{B}_i)}$$

$$h_{i+1} = \frac{-b}{x_{i+1} - p(\mathcal{B}_{i+1})}$$

$$h_i^{\min} \leq h_i \leq h_i^{\max}, h_{i+1}^{\min} \leq h_{i+1} \leq h_{i+1}^{\max}$$

$$p(\mathcal{B}_i) + w_i^{\min} \leq x_i \leq p(\mathcal{B}_i) + w_i^{\max},$$

$$p(\mathcal{B}_{i+1}) - w_{i+1}^{\max} \leq x_{i+1} \leq p(\mathcal{B}_{i+1}) - w_{i+1}^{\min}$$

We prove that the above inequalities have their minimum when the heights h_i and h_{i+1} are equal or when one of them is equal to one of its extreme values.

Definition 3.2 A good realization for two elastic rectangles $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$ is at a **boundary value** if it uses a rectangle from $\mathcal{B}_i$ or $\mathcal{B}_{i+1}$, or both, of maximum or minimum possible height.

Lemma 3.2 The good realization of minimum total height for two facing x -base rectangles (if it exists) either consists of two rectangles of equal height, or occurs at a boundary value.

PROOF. Take a good realization of minimum total height. Suppose the heights of the rectangles are h_1 and h_2 and suppose that $h_1 < h_2$. Thus the total height is h_2 , and something prevents us from decreasing h_2 . Either h_2 is at the minimum possible height, or the two rectangles touch and we cannot trade them off because h_1 is at the maximum possible height. In either case we are at a boundary value.

□

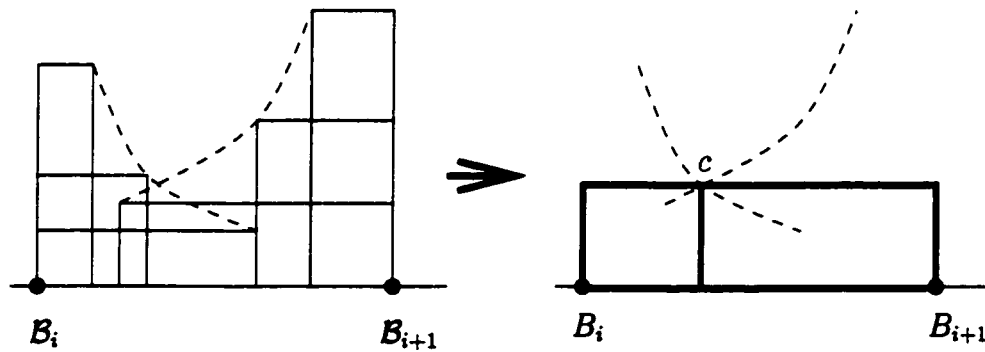


Figure 3.10: Elastic rectangles whose hyperbola segments intersect.

Consider the hyperbola segments $\psi(\mathcal{B}_i)$ and $\psi(\mathcal{B}_{i+1})$ described by the elastic rectangles. Let $\Psi(\mathcal{B}_i)$ and $\Psi(\mathcal{B}_{i+1})$ be the entire hyperbolas.

Compute the intersection point c of $\Psi(\mathcal{B}_i)$ and $\Psi(\mathcal{B}_{i+1})$ as follows. Assume that the areas of $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$ are a and b respectively. Then the equation of the hyperbola for $\mathcal{B}_i$ is $y = a/(x - p_i)$ and the one for $\mathcal{B}_{i+1}$ is $y = -b/(x - p_{i+1})$. Thus, by making these equations equal in their y -coordinates we obtain the x -coordinate of c equal to $(ap_{i+1} + bp_i)/(a + b)$ and we can obtain the corresponding y -coordinate.

If c is contained in the hyperbola segments $\psi(\mathcal{B}_i)$ and $\psi(\mathcal{B}_{i+1})$ then we select rectangles B_i and B_{i+1} with height equal to $y(c)$. Clearly, this gives the minimum height good realization for the two elastic rectangles. The right pane of Figure 3.10 shows the realizations chosen for elastic rectangles $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$.

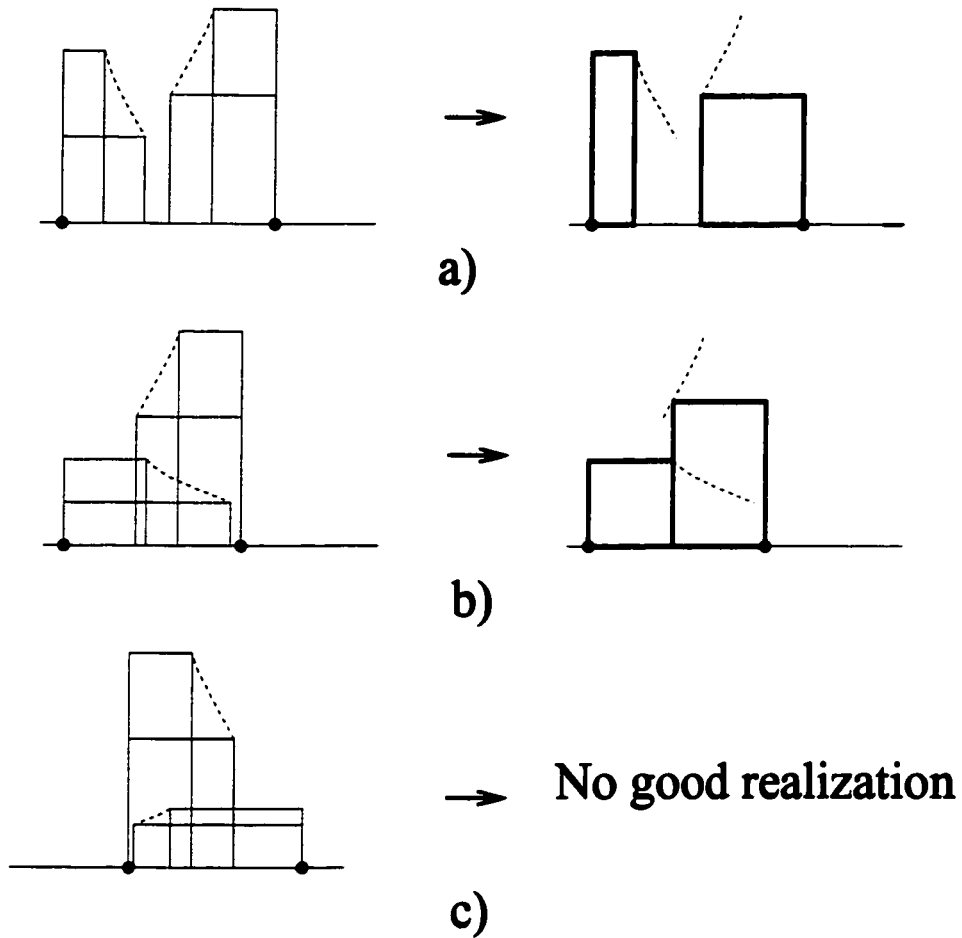
In the case when c is not contained in one, or both, of $\psi(\mathcal{B}_i)$ or $\psi(\mathcal{B}_{i+1})$, we must consider the four boundary values. Consider just one case. The others are similar. Suppose we take the maximum height realization of $\mathcal{B}_i$ and call it $B_i^{h_{\max}}$. Then, we look for the maximum width realization B_{i+1} of $\mathcal{B}_{i+1}$, not intersecting $B_i^{h_{\max}}$ (see Figures 3.11a and 3.11b), if such a realization exists. (See Figure 3.11c for an example where no such realization exists.)

After obtaining all the possible good realizations at the boundary values for $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$, from them we take the one of minimum height. By Lemma 3.2, this realization is one of minimum height among all possible good realizations for the set of elastic rectangles.

3.5.2 Elastic Rectangles in Opposite Axes

Here we study the slightly more general problem that arises in case II, subcase 2 of the algorithm, in which the left-hand elastic rectangle no longer lies on the x -axis. In this case we have an x -base elastic rectangle and a y -base elastic rectangle and we wish to find realizations of the two elastic rectangles with minimum total height—i.e. minimizing the maximum height of the two. This problem can be solved in exactly the same way as when the elastic rectangles lie in the same axis. As in Section 3.5.1 we have an analogous system of inequalities describing the problem.

Indeed, assume that we have an x -base rectangle, $\mathcal{B}$, with its anchor at its bottom-right corner, and a y -base rectangle, $\mathcal{L}$, with its anchor at its bottom-left corner. Let $\psi(\mathcal{B})$ and $\psi(\mathcal{L})$ be the hyperbola segments described by $\mathcal{B}$ and $\mathcal{L}$ respectively, and let $\Psi(\mathcal{B})$ and $\Psi(\mathcal{L})$ be the hyperbolas containing segments $\psi(\mathcal{B})$ and $\psi(\mathcal{L})$, respectively. Let c be the point where $\Psi(\mathcal{B})$ and $\Psi(\mathcal{L})$ intersect, and let $p = p(\mathcal{B})$ and $q = p(\mathcal{L})$.


 Figure 3.11: Realizations with boundary value $B_i^{h_{\max}}$.

Note that the width of a good realization of $\mathcal{B}$ and $\mathcal{L}$ is determined by p . Then, we aim to minimize the height of a good realization. If c is contained in the hyperbola segments $\psi(\mathcal{B})$ and $\psi(\mathcal{L})$ then we select a rectangle B with height equal to $y(c)$ and a rectangle L with width equal to $x(c)$. Clearly this gives the minimum-width good realization for those elastic rectangles (see Figure 3.12).

We obtain the point c as follows. Assume $\mathcal{B}$ has area a and anchor p , then its hyperbola equation is of the form $y = -a/(x-p)$. Assume $\mathcal{L}$ has area b and anchor q , then its hyperbola equation is $y = b/x + q$. In order to obtain the intersection

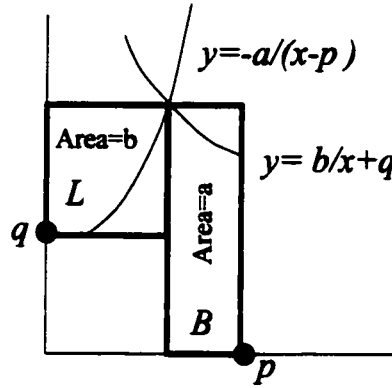


Figure 3.12: Elastic rectangles in opposite axes whose hyperbola segments intersect.

point of two hyperbolas we make the equations equal. Then, we solve the resulting quadratic expression $qx^2 + (a + b - pq)x - bp = 0$ by using the traditional quadratic formula to obtain the common x -coordinate.

For the case where c is not contained in one or both $\psi(\mathcal{B})$ and $\psi(\mathcal{L})$ we have the following lemma.

Lemma 3.3 *Given an x -base rectangle $\mathcal{B}$ with a bottom-right anchor and a y -base rectangle $\mathcal{L}$ with bottom-left anchor, described by hyperbola segments $\psi(\mathcal{B})$ and $\psi(\mathcal{L})$ respectively. If the intersection of their whole hyperbolas $\Psi(\mathcal{B})$ and $\Psi(\mathcal{L})$ is not contained in one or both $\psi(\mathcal{B})$ and $\psi(\mathcal{L})$, then the minimum height good realization, if it exists, is at a boundary value.*

The proof of this lemma is similar to the one of Lemma 3.2. Figure 3.13 depicts some examples of good realizations that are at boundary values.

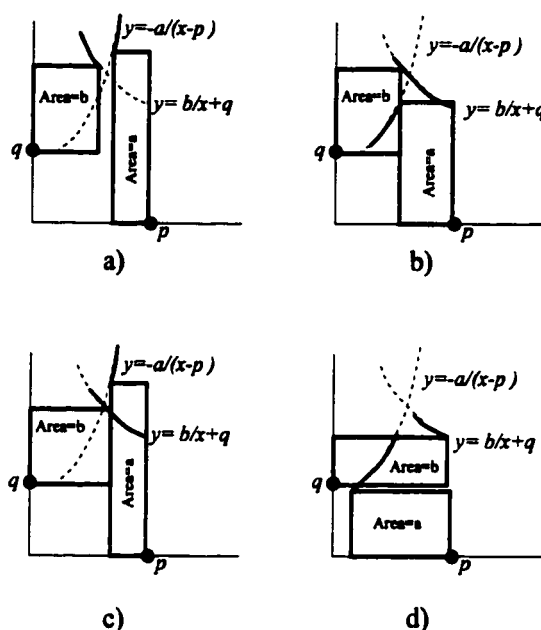


Figure 3.13: Elastic rectangles in opposite axes whose hyperbola segments do not intersect.

3.6 Discrete Two-Axis Labeling Problem

The most common use of a label is to write text in it, in which case we can use one, two or more rows for the text. In this case, the height of a label should not vary continuously, but take on one of a finite set of values. The *discrete two-axis labeling problem*, as its name indicates, is the two-axis elastic labeling problem having elastic rectangles with a discrete set of realizations. We call these elastic rectangles *discrete elastic rectangles*.

One thing to keep in mind is that when a paragraph is divided into individual lines, words cannot be broken arbitrarily. There is a detailed study of breaking paragraphs into lines done by Knuth and Plass [KP181]. Breaking a paragraph in

an appropriate way produces rectangles of roughly the same area, but not exactly the same area. To model this as a discrete labeling problem we obtain all possible rectangles that a paragraph produces and take the one with the biggest area. Then adjust the widths of the others to have the same area.

Note that a discrete elastic rectangle can be described by its anchor and the corners of its realizations opposite to the anchor. We call these corners *opposite anchors*.

In the two-axis algorithm the only part where we made use of the continuity of elastic rectangles was when a minimum total height or width good realization for two rectangles was obtained (see Section 3.5). Thus, we need to modify this part of the algorithm to obtain a solution for the discrete version of the problem.

We explain the case for two x -base rectangles. The case of one x -base rectangle and one y -base rectangle is similar.

Given two discrete elastic rectangles $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$, let $\Psi(\mathcal{B}_i)$ and $\Psi(\mathcal{B}_{i+1})$ be the whole hyperbolas containing the opposite anchors describing $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$ respectively. Let c be the intersection of $\Psi(\mathcal{B}_i)$ and $\Psi(\mathcal{B}_{i+1})$.

If $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$ have a common corner point then we select the realizations having this common point. Clearly this common point is the intersection of the whole hyperbolas $\Psi(\mathcal{B}_i)$ and $\Psi(\mathcal{B}_{i+1})$, moreover, the good realization thus obtained is of minimum height.

Now, if $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$ do not have a common point we decompose their realizations as follows:

1. all realizations of $\mathcal{B}_i$ and of $\mathcal{B}_{i+1}$ with opposite anchors above or equal to c ,

2. all realizations of $\mathcal{B}_i$ with opposite anchors above or equal to c , and all realizations of $\mathcal{B}_{i+1}$ with opposite anchors below or equal to c ,
3. all realizations of $\mathcal{B}_{i+1}$ with opposite anchors above or equal to c , and all realizations of $\mathcal{B}_i$ with opposite anchors below or equal to c .

Some of the above decompositions may be empty if one of $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$ have opposite anchors completely above or below c . Note that we did not consider all realizations of $\mathcal{B}_i$ and of $\mathcal{B}_{i+1}$ which are below c , since there does not exist a good realization using them.

Thus, for each decomposition we obtain a good realization of minimum height for the subset of considered realizations of $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$. From these realizations we take one with the minimum value for the minimum total heights. We prove that this good realization is of minimum total height for elastic rectangles $\mathcal{B}_i$ and $\mathcal{B}_{i+1}$.

To obtain a good realization of minimum total height for each decomposition we obtain the good realizations at boundary values of the elastic rectangles built with the realizations considered in the decomposition. From them we take the one with minimum total height value.

Lemma 3.4 *A good realization of minimum total height for two facing x -base discrete rectangles (if it exists) either consists of two rectangles of equal height, or occurs at a boundary value of the elastic rectangles contained in the subset of realizations considered in each decomposition.*

PROOF. If the elastic rectangles have a common opposite anchor then we have a good realization with rectangles of equal height. It is clear that if we decrease the height of one of those rectangles the height of the other must increase. Thus, this realization is of minimum total height.

If the elastic rectangles do not contain a common opposite anchor then a good realization with minimum total value for the elastic rectangles must be one of the good realizations of minimum total height of their decompositions. Suppose the height of the rectangles in a decomposition are h_1 and h_2 with $h_1 < h_2$. Then the total height is h_2 and something prevents us from decreasing h_2 . Either h_2 is at the minimum height in the decomposition or we cannot trade them off because h_1 is at the maximum possible height in the decomposition. Thus in either case we are at a boundary value. This occurs for each decomposition. $\square$

We can see examples to obtain a good realization of minimum height in Figure 3.14. In this figure the gray dots indicate the opposite anchors that a discrete elastic rectangle has and the dashed curve is the hyperbola that contains such opposite anchors.

By replacing the way we obtain a good realization of minimum height or width as indicated above in the two-axis algorithm, we have an algorithm for the discrete two-axis labeling problem. We call the algorithm the discrete two-axis algorithm.

From the discussion given above the following theorem holds.

Theorem 3.3 *Given an instance of the discrete two-axis labeling problem the discrete two-axis algorithm finds a good realization if one exists.*

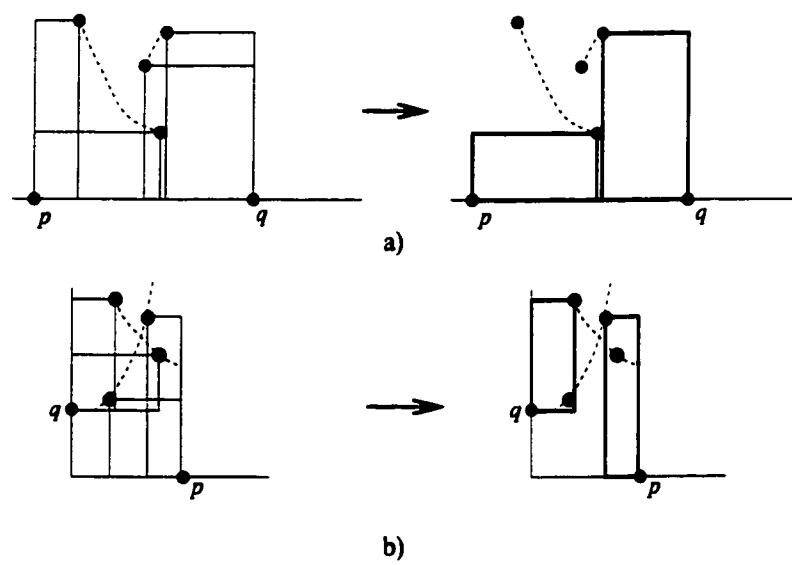


Figure 3.14: Good realizations for instances of the discrete two-axis labeling problem.

Chapter 4

Two-Parallel Lines Labeling Problem

In this chapter we study another case of the one-corner labeling problem, when the anchors of the elastic rectangles lie on two vertical lines, the *two-parallel lines problem*. In the first section, we define the problem. In the second one, we present an algorithm to solve the problem.

4.1 Definitions and Preliminaries

Given a vertical line we say that a one-corner elastic rectangle $\mathcal{E}$ is a *left-base* rectangle if its anchor lies in the line with position $Q(\mathcal{E}) = \{bl\}$, or $Q(\mathcal{E}) = \{tl\}$; and we say that it is a *right-base* rectangle if its anchor lies in the line with position $Q(\mathcal{E}) = \{tr\}$ or $Q(\mathcal{E}) = \{br\}$.

Definition 4.1 *Given two vertical lines and a set of left-base rectangles and right-base rectangles all rectangles located between the lines and with their anchors lying*

on the vertical lines the **two-parallel lines labeling problem** is to find a good realization for the union of these sets.

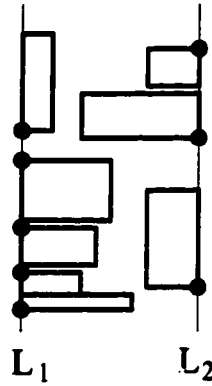


Figure 4.1: A good realization for an instance of the two-parallel lines labeling problem.

Let L_1 be the vertical line on the left and L_2 be the vertical line on the right. Figure 4.1 shows a good realization for an instance of the two-parallel lines labeling problem.

Our goal is to select a realization for each elastic rectangle such that together they form a good realization for the entire set. As in the previous chapter, we say that we fix an elastic rectangle when we select a realization from its family. We also indicate fixed rectangles using bold rectangles in our figures.

Fixing an elastic rectangle can affect our choices for other elastic rectangles. For example, when two consecutive anchors along a line have their rectangles facing each other, as in Figure 4.2a, we can trade off the height of one against the height of the other, and which solution is best depends on the particular elastic rectangles we have and in particular those on the other line. Similarly for the case when we have an anchor in each line with elastic rectangles facing each other (see Figure 4.2b).

Since we have elastic rectangles between two vertical lines, we can have facing rectangles in each line **and** facing each other between lines as shown in Figure 4.2c.

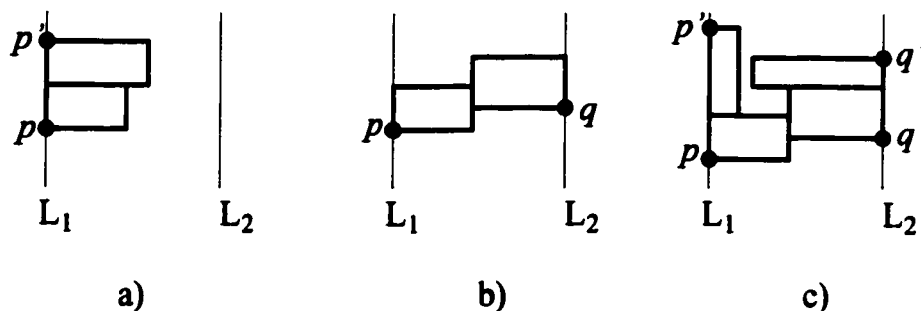


Figure 4.2: Rectangles facing each other.

Thus, once we fix a rectangle we want to ensure that if a good realization exists for the entire set, that we are still able to find it. That is, by fixing a rectangle we do not want to leave out all realizations of other elastic rectangles that will lead us to obtaining a good realization. In the following section, we present an algorithm that gives us such a guarantee.

4.2 An Algorithm for the Two-Parallel Lines Case

We use a greedy approach, adding elastic rectangles from bottom to top. We keep the most recently added rectangles elastic, and fix the earlier ones.

The idea of the algorithm is to build a good realization from the bottom to the top by adding elastic rectangles one by one. The algorithm will allow three of the added rectangles (including the two most recently added) to remain elastic; the others will be fixed. We will begin by describing the bottom-to-top ordering of the anchors in more detail; then we discuss the structure of a general step in the algorithm.

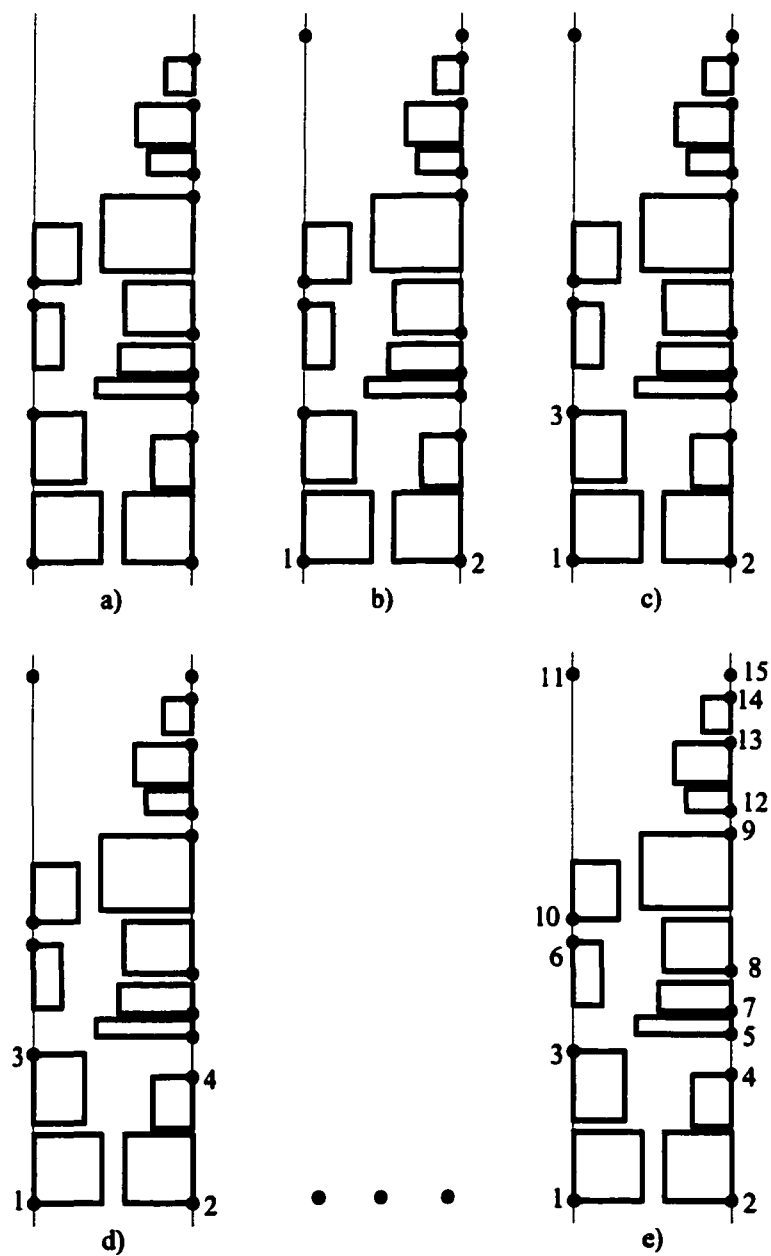


Figure 4.3: Smallest-moves-up ordering.

We consider the anchors in each line L_1 and L_2 in order from bottom to top. These two orderings are merged as follows: Initially, let p and q be the bottommost anchors of lines L_1 and L_2 , respectively —we take them in either order. More generally, let p and q be the most recently considered anchors of L_1 and L_2 , respectively. If p is below q we add the next anchor above p on L_1 . Otherwise, we add the next anchor above q on L_2 . If p and q have the same y -coordinate, we make either choice. We call this ordering the *smallest-moves-up ordering*.

For simplicity, we will add two dummy anchors with associated 0×0 rectangles at the tops of the lines L_1 and L_2 to ensure that all anchors get considered in this ordering. For example, consider a good realization of an instance of the two-parallel lines problem as shown in Figure 4.3a. In this figure, we take the bottommost anchors in each line and enumerate them as 1 and 2 (see Figure 4.3b); since they have the same y -coordinate the next rectangle to be considered could be above either of them. We add to the ordering the one above 1 (see Figure 4.3c). Now the most recently considered anchors are 2 and 3, with 2 below 3, thus we add to the ordering the anchor above 2. The recently considered anchors now are 3 and 4 (see Figure 4.3d), with 4 below 3, thus we add the rectangle above 4. We continue with this procedure until all anchors have been considered and we obtain the ordering shown in Figure 4.3e.

In general, as the algorithm proceeds, three of the elastic rectangles that have been considered remain elastic, and the others are fixed. If r and s are the current topmost anchors of L_1 and L_2 (not necessarily respectively), and $y(r) \geq y(s)$ and t is the anchor below r on its line, then the rectangles that remain elastic are those associated with r , s , and t . Because r was added to the ordering to increase the smallest y -coordinate, thus $y(t) \leq y(s)$, and the points form what we call a *topmost C-shape*. To be specific, r, s, t form a top-most *C-shape* if r and t are consecutive

on one line, s on the other, $y(r) \geq y(s) \geq y(t)$. We call r , s , and t the top, middle, and bottom points of the C -shape respectively.

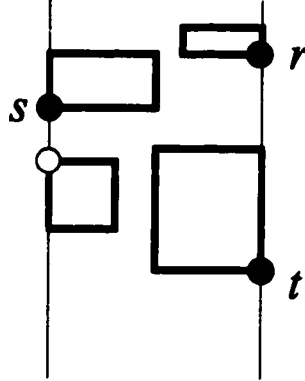


Figure 4.4: C -shape.

In Figure 4.4, the anchors in bold are in a C -shape. Note that the anchors that form the topmost C -shape are not necessarily the three topmost anchors nor the three most recently added anchors—in particular, though r and s are topmost, t may not be (see the anchor indicated with the hollow dot in Figure 4.4).

The idea is to leave elastic only those rectangles with anchors in the topmost C -shape of the realization obtained so far. In this way, when a rectangle is added to the realization, we consider the three rectangles in this C -shape together with the elastic rectangle that is added to the realization, following the smallest-moves-up order. Taking into consideration these four elastic rectangles we fix the rectangle whose anchor is no longer in the topmost C -shape. Later on the correctness of this approach will be proven. In the following section, we focus on these four elastic rectangles.

4.2.1 Fixing a Rectangle

We have three elastic rectangles with anchors in a C -shape and the elastic rectangle added as indicated in the section above. The anchor of the rectangle added has to be above of the middle point of the topmost C -shape. We will fix the rectangle whose anchor leaves the topmost C -shape. This rectangle may limit the elasticity of the other rectangles. We must ensure that we do not exclude a good realization for the whole set of elastic rectangles if one exists, so we have to keep in mind the global picture.

Let $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$ be the three elastic rectangles with anchors, r , s , and t respectively forming the topmost C -shape (see Figure 4.5).

- Topmost C -shape
- Rectangle to be fixed

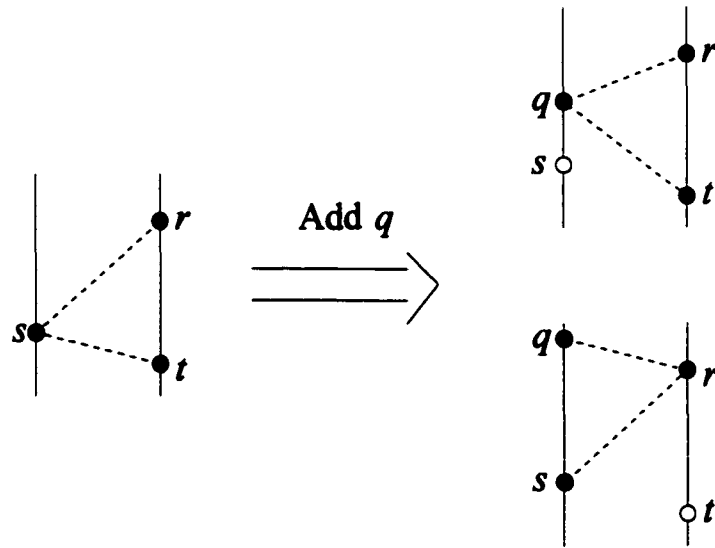


Figure 4.5: Adding anchor q to the topmost C -shape.

Consider what happens when we add one more elastic rectangle. Its anchor,

q , enters the topmost C -shape. Another one leaves, and we must fix its elastic rectangle. Since the anchor q must be next in smallest-moves-up ordering it must be above s on its line. The new topmost C -shape depends on the position of q : If q lies below r then the new topmost C -shape consists of r, q, t and we must fix the elastic rectangle at s . If q lies above r then the new topmost C -shape consists of q, r, s and we must fix the elastic rectangle at t . See Figure 4.5.

We will find it notationally more convenient to have one symbol for the elastic rectangle that leaves the topmost C -shape. We will therefore treat the *previous* step of the algorithm, the one where r, s, t became the topmost C -shape. Note that by definition of a C -shape that $y(r) \geq y(s) \geq y(t)$. Let u be the anchor that just left the topmost C -shape. Then u lies under s in its line. If u is above t then s is the newly added point and r, u, t is the old C -shape (see Figure 4.6a); and if u is below t then r is the newly added point and s, t, u is the old C -shape (see Figure 4.6b).

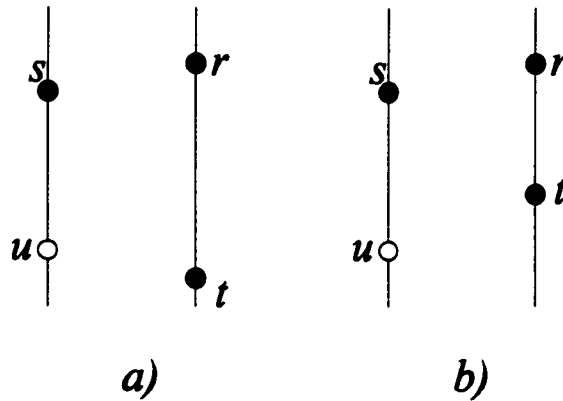


Figure 4.6: Anchor u leaves the topmost C -shape.

We must fix $\mathcal{U}$, the elastic rectangle with anchor u . A case-by-case analysis seems necessary. We must consider whether each anchor is top or bottom. We will show that we can maintain the structure that the middle and bottom anchors of the topmost C -shape are bottom anchors or else top anchors of fixed rectangles.

Therefore, we can assume that u and t are bottom anchors.

On the other hand, in order to maintain this structure, we must now be sure to fix S if s is a top anchor. We will treat the four cases where r and s are top/bottom anchors.

We note here, and later will take for granted, that in choosing realizations for elastic rectangles, we must take care to avoid the rectangles that have already been fixed. Then, when fixing a rectangle C to realization C we say that we “update an elastic rectangle $\mathcal{E}$ with respect to a realization C ” when we take away the realizations of $\mathcal{E}$ intersecting C .

We can ignore the relative vertical order of u and t because of the following simplification which we can apply to any topmost C -shape.

Lemma 4.1 *Suppose we have a topmost C -shape where the middle and bottom points are bottom anchors of their elastic rectangles or top anchors of fixed rectangles. If the minimum height realization with respect to the fixed rectangles, U , of the bottom-most elastic rectangle, $\mathcal{U}$, of the C -shape lies below the middle anchor of the C -shape, then we can fix $\mathcal{U}$ to U without precluding a good realization for the whole set.*

PROOF. We show that if there is a good realization for the whole set that uses the rectangles fixed up to now, then there is a good realization for the whole set using the rectangles fixed up to now and using U . Let s, t, u be the anchors of the topmost C -shape. Consider a good realization for the whole set using the rectangles fixed up to now. Let S', T' , and U' be the realizations for the elastic rectangles at s, t, u respectively. Then, U' is taller than or equal to U . Hence, since U' does not intersect S' neither does U . Also since U is below t , it does not intersect T' . Thus,

we can replace U' by U and preserve a good realization. (Figure 4.7 illustrates this case, the fixed rectangle is indicated in bold.) $\square$

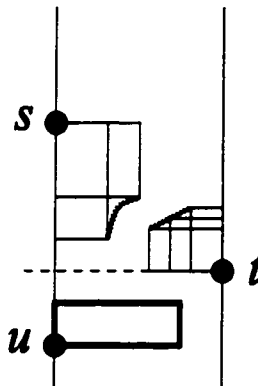


Figure 4.7: Fixing an elastic rectangle to a realization lying below t .

Applied to our situation, this simplification implies that the realizations of the elastic rectangles $\mathcal{U}$ and $\mathcal{T}$ must lie side by side, and the relative ordering of u and t is irrelevant. In the figures we will draw u and t at the same y -coordinate.

Now, we concentrate on finding a good realization for $\mathcal{R}$, $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{U}$ while keeping in mind that we could have elastic rectangles above them. Let $\mathcal{A}$ be the set of elastic rectangles that are above $\mathcal{R}$, $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{U}$. When fixing an elastic rectangle the realizations intersecting it are discarded. Thus, we may limit the elasticity of other elastic rectangles. We proceed to describe which good realization for $\mathcal{R}$, $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{U}$ limits as little as possible the choices for a good realization of the whole set. We explain later, in Section 4.2.3, how to find such realizations. For now we assume that we are able to find them.

In all cases we will concentrate on the horizontal line, l_s , going through the point s because we now have full information about the two, three, or four elastic rectangles that have realizations intersecting the region below this line. We call these the elastic rectangles that are *active below* l_s . We will say that two elastic

rectangles *interact* if they have realizations that intersect. In all cases we must fix $\mathcal{U}$. Its realization must lie below l_s and so—of all the rectangles not yet fixed—it may interact only with the other one, two, or three elastic rectangles that are active below l_s . Because some of these elastic rectangles may interact with as-yet-unseen elastic rectangles, we must fix $\mathcal{U}$ in such a way that we limit as little as possible the choices for these other elastic rectangles.

Now, we proceed to present the cases to obtain a good realization for $\mathcal{R}$, $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{U}$ when r and s are bottom/top anchors and u and t are bottom anchors.

Case 1. Both points r and s are bottom anchors. Figure 4.8 shows a good realization of an instance of this case. We must fix $\mathcal{U}$.

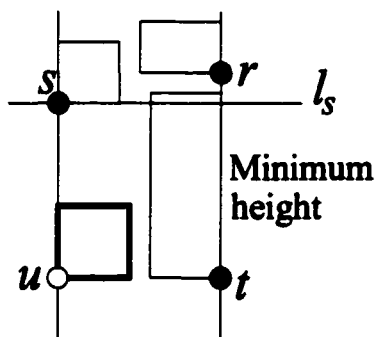


Figure 4.8: Fixing $\mathcal{U}$ for Case 1.

Elastic rectangle $\mathcal{U}$ interacts only with $\mathcal{T}$, since no other elastic rectangle is active below l_s . In this case, we want to preserve $\mathcal{T}$ “as elastic as possible”, that is we leave it with the biggest set of realizations possible. Then we obtain a good realization $\mathcal{U}$ and $\mathcal{T}$ that minimizes the height of $\mathcal{T}$. It is clear that we are leaving out of $\mathcal{T}$ the smallest possible set of realizations that do not form a good realization with $\mathcal{U}$. Because the realizations of $\mathcal{U}$ and $\mathcal{T}$ must lie side to side we can clearly fix $\mathcal{U}$ to its realization of minimum width.

Claim 4.1 *If there exists a good realization G for elastic rectangles $\mathcal{R}$, $\mathcal{S}$, $\mathcal{T}$, $\mathcal{U}$, and set $\mathcal{A}$, then we can replace the realization of $\mathcal{U}$ by U described above and still have a good realization.*

PROOF. Let G be a good realization for $\mathcal{R}$, $\mathcal{S}$, $\mathcal{T}$, $\mathcal{U}$, and set $\mathcal{A}$. Let R' , S' , T' , and U' be the realizations in G for $\mathcal{R}$, $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{U}$, respectively. We have that U' is wider than or equal to U . Thus, if U' does not intersect T' , U does not either. Therefore, we can replace U' by U and preserve a good realization. $\square$

Case 2. Point r is bottom anchor and s is top anchor. Figure 4.9 shows a good realization of an instance for this case. We must fix $\mathcal{U}$ and $\mathcal{S}$.

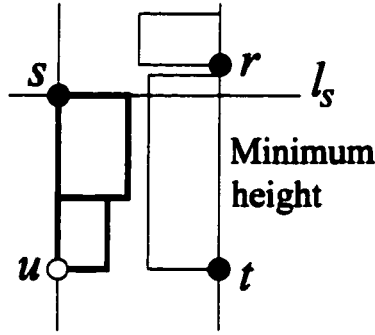


Figure 4.9: Fixing $\mathcal{U}$ and $\mathcal{S}$ for Case 2.

Elastic rectangle $\mathcal{U}$ interacts only with $\mathcal{T}$ and $\mathcal{S}$, since no other elastic rectangle is active below l_s . In this case, any realization of $\mathcal{S}$ is below l_s , therefore we fix $\mathcal{S}$ as well.

We want to find a good realization that leaves $T \in \mathcal{T}$, $U \in \mathcal{U}$, and $S \in \mathcal{S}$ below l_s . If we can find this realization we fix $\mathcal{T}$, $\mathcal{U}$ and $\mathcal{S}$ to T , U and S . Elastic rectangle $\mathcal{T}$ can be fixed to T by applying Lemma 4.1.

If we cannot find the above realization then in any good realization $\mathcal{T}$ must stick up above line l_s . In this case, we want to keep $\mathcal{T}$ as elastic as possible. Thus,

we find a realization that minimizes the height of $\mathcal{T}$. We can obtain this realization by fixing $\mathcal{U}$ and $\mathcal{S}$ to their realizations of minimum total width.

Claim 4.2 *If there exists a good realization G for elastic rectangles $\mathcal{R}$, $\mathcal{S}$, $\mathcal{T}$, $\mathcal{U}$, and set $\mathcal{A}$, then we can replace the realizations of $\mathcal{U}$ and $\mathcal{S}$ by U and S described above and still have a good realization.*

PROOF. If we have a realization $U, S, T \in \mathcal{T}$ below l_s , we replace them in G and maintain a good realization with $\mathcal{R}$ and $\mathcal{A}$. If we do not have this realization then the realization of $\mathcal{T}$ in G must stick up above l_s , and thus must lie beside the realizations of $\mathcal{U}$ and $\mathcal{S}$. Thus, replacing the realizations of $\mathcal{U}$ and $\mathcal{S}$ by U, S , the ones of minimum total width, leaves a good realization with $\mathcal{R}$ and $\mathcal{A}$. $\square$

Case 3. Point r is top anchor and s is bottom anchor. Figure 4.10 shows a good realization for an instance of this case. We must fix $\mathcal{U}$.

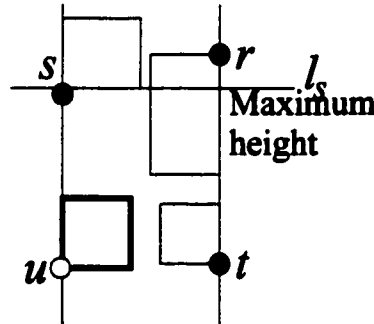


Figure 4.10: Fixing $\mathcal{U}$ for Case 3.

Elastic rectangle $\mathcal{U}$ interacts only with $\mathcal{T}$ and $\mathcal{R}$ since no other rectangle is active below l_s . Thus, we aim to obtain for a good realization R, T , and U of $\mathcal{R}$, $\mathcal{T}$, and $\mathcal{U}$ that leaves U and T below line l_s .

If we cannot obtain such a realization it is because $\mathcal{T}$ must stick up above line l_s . Then, we can apply the same arguments as in Case 1 and fix $\mathcal{U}$ to its realization

of minimum width. As in Case 1 we can prove that this does not preclude a good realization for the whole set.

Now, we concentrate on the case where there is a good realization for $\mathcal{U}$ and $\mathcal{T}$ lying below line l_s . Not all such realizations are equally good. Since anchor r is above l_s , then $\mathcal{R}$ may interact with $\mathcal{S}$ and with elastic rectangles in $\mathcal{A}$. Our objective is to leave $\mathcal{R}$ with a set of realizations that will lead us to a good realization with $\mathcal{S}$ and $\mathcal{A}$, if one exists.

We find a realization $R \in \mathcal{R}$ of maximum height that permits a good realization U, T of $\mathcal{U}$ and $\mathcal{T}$. We fix $\mathcal{U}$ to U . We claim that this does not preclude a realization for the whole set.

Claim 4.3 *If there exists a good realization G for elastic rectangles $\mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{U}$, and $\mathcal{A}$, then we can replace the realization of $\mathcal{U}$ by U described above and still have a good realization.*

PROOF. Let G be a good realization for $\mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{U}$, and $\mathcal{A}$. Let R', S', T' , and U' be the realizations in G for $\mathcal{R}, \mathcal{S}, \mathcal{T}$, and $\mathcal{U}$, respectively.

Either R' sticks down below line l_s or not. If it does, then we replace realizations U', T' and R' by U, T , and R obtained above. Since R is taller and thinner than or equal to R' , thus R does not interfere with any other rectangle in G above line l_s .

If R' does not stick down below line l_s , then we have two cases. Realizations U and T obtained above lie below line l_s or not. If U and T obtained above lie below l_s , we can replace U' and T' by U and T and preserve a good realization. Otherwise, T sticks up above line l_s , then the proof is analogous to the one of Claim 4.1 for Case 1. □

Case 4. Both points r and s are top anchors. Figure 4.11 shows a good realization for an instance of this case. We must fix $\mathcal{U}$ and $\mathcal{S}$.

We can see that this case is similar to Case 3. Instead of looking for a good realization of $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{U}$, here we look for a good realization of $\mathcal{R}$, $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{U}$ since elastic rectangle $\mathcal{U}$ interacts with $\mathcal{R}$, $\mathcal{S}$, and $\mathcal{T}$.

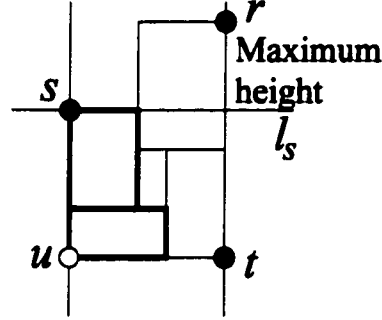


Figure 4.11: Fixing $\mathcal{U}$ and $\mathcal{S}$ for Case 4.

First, we aim to obtain a good realization R, S, T, U of $\mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{U}$ that leaves S, T, U below line l_s . If we cannot obtain such a realization and a good realization exists it is because $\mathcal{T}$ must stick up above line l_s . Then, we can apply the same arguments as in Case 2 and fix $\mathcal{S}$ and $\mathcal{U}$ to their realizations of minimum total width.

To obtain a good realization with S, T, U lying below line l_s , as we saw in Case 3 not all realizations are equally good since $\mathcal{R}$ may interact with elastic rectangles in $\mathcal{A}$. In this case we find a realization $R \in \mathcal{R}$ of maximum height that permits a good realization S, T, U of $\mathcal{S}, \mathcal{T}, \mathcal{U}$. We fix $\mathcal{U}$ to U and $\mathcal{S}$ to S .

Note that fixing $\mathcal{S}$ and $\mathcal{T}$ to their realizations of minimum total width may not maximize the height of $\mathcal{R}$ for a good realization. In Figure 4.12 we obtained a good realization for the elastic rectangles $\mathcal{R}, \mathcal{S}, \mathcal{T}$, and $\mathcal{U}$ corresponding to the ones of Figure 4.11. In Figure 4.12 the height of $\mathcal{R}$ can be maximized by making S thinner

and in consequence U wider, thus the total width of S and U is not minimum but the height of $\mathcal{R}$ has increased as shown in Figure 4.11. In Section 4.2.3 we explain in detail how to obtain a realization $R \in \mathcal{R}$ of maximum height that permits a good realization U, S, T of $\mathcal{U}, \mathcal{S}, \mathcal{T}$. For now, we assume that we can obtain such realization.

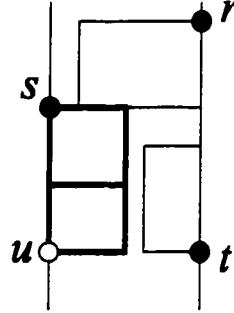


Figure 4.12: A realization for $\mathcal{U}$ and $\mathcal{S}$ of minimum total width.

Claim 4.4 *If there exists a good realization G for elastic rectangles $\mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{U}$, and set $\mathcal{A}$, then we can replace the realizations of $\mathcal{U}$ and $\mathcal{S}$ by U and S described above and still have a good realization.*

PROOF. Let R', S', T' , and U' be the realizations in G for $\mathcal{R}, \mathcal{S}, \mathcal{T}$, and $\mathcal{U}$, respectively. We have that either R' sticks down below line l_s or not. If it does then we replace realizations R', S', T' , and U' by R, S, T , and U obtained above. Since R is taller and thinner than or equal to R' , thus R does not interfere with any of the rectangles of $\mathcal{A}$.

If R' does not stick down below line l_s , then we have two cases. Realizations S, T, U obtained above lie below line l_s or not. If they lie below line l_s , then we can replace S', T', U' by S, T, U and preserve a good realization. Otherwise, we have

that T sticks up above line l_s , then the proof is equal to the one of Claim 4.2 for Case 2. $\square$

Figure 4.13 illustrates together the four cases for fixing the elastic rectangle $\mathcal{U}$ that leaves the topmost C -shape. Note that the top and bottom pictures for each case show the possible relative positions for u and t .

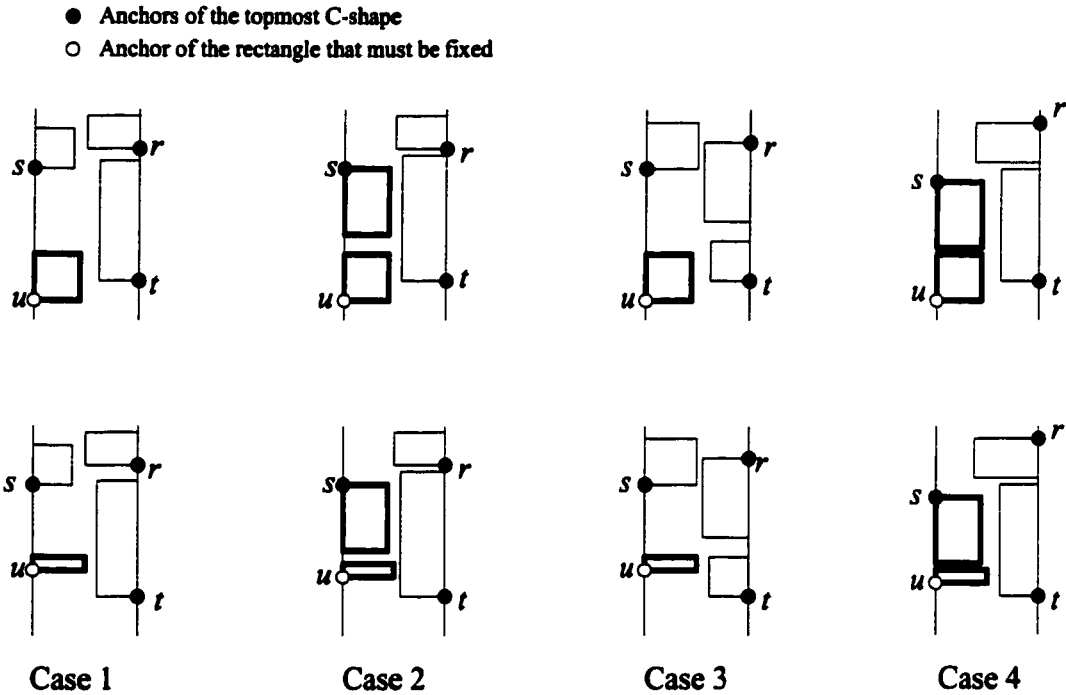


Figure 4.13: Cases of a realization when r , s , and t are in the topmost C -shape.

4.2.2 Algorithm

We build a good realization from the bottom to the top by adding elastic rectangles one by one and updating the good realization. The rectangles are added in smallest-moves-up order. Each time an elastic rectangle is added to the realization we obtain a good realization for the three elastic rectangles whose anchors are in the topmost

C-shape and the rectangle that is added. To obtain this good realization we use the criteria of Section 4.2.1. To use these arguments we need that the bottommost elastic rectangles in each line have their anchors as bottom corners. In order to start with this configuration we place at the bottom of each line two dummy 0×0 rectangles, that do not intersect any other rectangle. As discussed before, we also place two dummy 0×0 rectangles at the top of each line to guarantee that all the rectangles will be enumerated in smallest-moves-up order. We present our algorithm below.

Two-parallel lines algorithm:

1. Add dummy 0×0 rectangles at the bottom of each line.
2. Add dummy 0×0 rectangles at the top of each line.
3. Obtain a smallest-moves-up order of the anchors.
4. Initialize the realization with the first 3 rectangles in the ordering. These form a *C*-shape, and all remain elastic.
5. While not all elastic rectangles have been added to the realization do
 - (a) Add the next elastic rectangle in the smallest-moves-up order to the realization.
 - (b) Find a good realization for the elastic rectangles with anchors in the topmost *C*-shape and the rectangle added using the cases from Section 4.2.1. If this realization does not exist, then stop and report no solution. If a good realization is obtained fix and update the elastic rectangles as described in Section 4.2.1.

The algorithm described above takes $O(n \log n)$ time, where $n = n_L + n_R$ and n_L is the number of left-base rectangles in L_1 and n_R is the number of right-base rectangles in L_2 . Sorting the anchors takes $O(n \log n)$ time. The process of fixing a rectangle involves intersecting hyperbolas of at most four elastic rectangles which takes constant time for a real-RAM model of computation. We repeat the process of fixing a rectangle n times.

Theorem 4.1 *There is an algorithm for the two-parallel lines labeling problem with running time $O(n)$ where n is the number of points, and we assume they are given in sorted order along the two lines.*

See Section 4.2.3 for details. The proof of this theorem follows easily using Claims 4.1, 4.2, 4.3, and 4.4 in a proof by induction.

4.2.3 Hyperbolas

After we determined which criterion to use in Section 4.2.1 to get a good realization for elastic rectangles $\mathcal{U}$, $\mathcal{S}$, $\mathcal{T}$, and $\mathcal{R}$, we explain in this section the actual computation of this good realization. For example, in case 2 we need to find a good realization of $\mathcal{U}$, $\mathcal{T}$, and $\mathcal{S}$ below line l_s , or else to minimize the total width of the realization of $\mathcal{U}$ and $\mathcal{S}$. In this section we explain exactly how we can obtain these good realizations.

Observe that the algorithm does not actually rely on the fact that the area of an elastic label is constant, but only on the fact that the family of rectangles is captured by a hyperbola segment tracing the locus of the corner of the rectangle opposite the anchor (this fact will be of use in Chapter 5).

Remember that we assume that all realizations of $\mathcal{U}$ stick up above anchor t . Otherwise, we fix $\mathcal{U}$ to be of minimum height and we fix the other rectangles according to the required properties without worrying about taking into consideration rectangle $\mathcal{U}$.

We proceed to compute a good realization for each of the cases described in Section 4.2.1. Recall that u, s, t, r are the anchors of $\mathcal{U}, \mathcal{S}, \mathcal{T}$, and $\mathcal{R}$ respectively, with s, t, r the topmost C -shape and u below s , and l_s is the horizontal line passing through anchor s .

Case 1. Both points r and s are bottom anchors. (See Figure 4.8.) Here we only need to set $U \in \mathcal{U}$ to its realization of minimum width below l_s .

Case 2. Point r is bottom anchor and s is top anchor. (See Figure 4.9.) We first try to find a good realization with $U, \mathcal{S}, \mathcal{T}$ below l_s . There are two possible scenarios depending on whether the horizontal line through the bottom of $\mathcal{S}$ misses or intersects the interior of $\mathcal{T}$. In the first scenario $\mathcal{S}$ lies above $\mathcal{T}$ (see Figure 4.14a) and in the second it lies beside $\mathcal{T}$ (see Figure 4.14b).

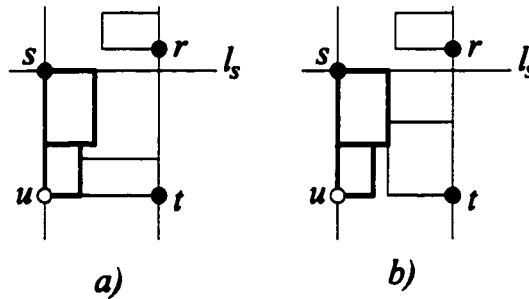


Figure 4.14: Scenarios for a good realization.

For the first scenario, $\mathcal{S}$ above both $\mathcal{U}$ and $\mathcal{T}$, we compute the realization of $\mathcal{U}$ and $\mathcal{T}$ of minimum total height. This means intersecting two hyperbolas, (as the anchors of $\mathcal{U}$ and $\mathcal{T}$ may not lie in a horizontal line) then we obtain a good

realization of minimum total height for them as indicated in Section 3.5.2 (the two-axis case having elastic rectangles in opposite axis).

For the second scenario, T lies beside S and U , we obtain the realization of S and U of minimum total width as indicated in Section 3.5.1.

If we find a good realization with U, S, T below l_s , then we fix $\mathcal{U}, \mathcal{S}, \mathcal{T}$ to them. Otherwise, we fix $\mathcal{U}$ and $\mathcal{S}$ to the realizations U and S of minimum total width, obtained as indicated in Section 3.5.1.

Case 3. Point r is top anchor and s is bottom anchor. (See Figure 4.10.) We first try to find a good realization with U and T below l_s . We obtain the realizations of minimum total height for $\mathcal{U}$ and $\mathcal{T}$, as indicated in Section 3.5.2. If these realizations do not lie below l_s , then we fix $\mathcal{U}$ to its realization of minimum width. Otherwise, we want to maximize the height of R . There are two possible scenarios depending on whether the horizontal line through the bottom of R intersects or misses the interior of U . In the first scenario R lies beside U , and in the second it lies above U . In the first scenario, we fix $\mathcal{U}$ to its realization U of minimum width, fix $\mathcal{T}$ to its realization T of minimum height that does not intersect U , and fix $\mathcal{R}$ to its realization R of maximum height not intersecting U and T . In the second, we obtain a good realization U, T for $\mathcal{U}, \mathcal{T}$ of minimum total height, as indicated in Section 3.5.2, and fix $\mathcal{R}$ to its realization of maximum height that does not intersect U and T .

Case 4. Both points r and s are top anchors. (See Figure 4.11.) This case is more complicated than the first 3 cases in that we have four elastic rectangles interacting with each other, and cannot always reduce, as we did above, to optimizing one or two at a time.

First we try to find a good realization U, S, T for $\mathcal{U}, \mathcal{S}, \mathcal{T}$, below l_s . There are

two scenarios: S is above T or S is beside T . To find this realization we proceed as in Case 2. If we cannot find such a realization then we fix $\mathcal{U}$ and $\mathcal{S}$ to their realizations of minimum total width as indicated in Section 3.5.1, and update the elasticity of $\mathcal{T}$. Otherwise we need to maximize the height of R . If R lies above l_s , then any good realization of $\mathcal{U}$, $\mathcal{S}$, $\mathcal{T}$ below l_s is fine. The difficulty of case 4 arrives when R sticks down below l_s , i.e. when R is beside S . We proceed to explain this case.

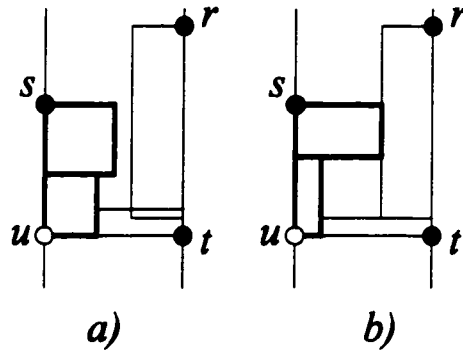


Figure 4.15: a) A realization having $\mathcal{U}$ and $\mathcal{S}$ of minimum total width, and $\mathcal{U}$, $\mathcal{T}$ of minimum total height. b) A good realization for a).

We show in Figure 4.15a that obtaining a good realization $\mathcal{U}, \mathcal{S}$ of minimum total width, or $\mathcal{U}, \mathcal{T}$ of minimum total height may not permit a good realization, while there is a good realization for this scenario as shown in Figure 4.15b.

We distinguish three scenarios: either there is a horizontal line separating S and R above from U and T below, or there is a horizontal line through the interiors of S and T , or there is a horizontal line through the interiors of S and R .

In the first scenario we find a realization of minimum total height for U and T , and then realizations for R and S of maximum possible height disjoint from U and T .

In the second scenario T lies beside S and U , so we find a realization of minimum total width for S and U , and then realizations for T and R of minimum and maximum height respectively disjoint from S and U .

In the third scenario we will concentrate on two possible types of solution. An *extreme* solution has R of the maximum height in its range, or T of minimum height, or U of maximum height, or S of minimum height. A *grid lock* solution has the bottom side of R touching the top side of T , the left side of T touching the right side of U , the top side of U touching the bottom side of S , and the right side of S touching the left side of R .

Claim 4.5 *If there is a solution in the third scenario there is either an extreme solution or a grid-lock solution.*

PROOF. Take a solution. Recall that this means a good realization R, S, T, U , with R of maximum height among all such good realizations. Recall that in the third scenario there is a horizontal line crossing the interiors of U and R . See Figure 14.15(b). If R is not at the maximum height in its range then its bottom must touch T . If T is not at the maximum width in its range (which is the same as being at the minimum height in its range) then its left side must touch U . If U is not at the maximum height in its range then its top must touch S . And if S is not at the maximum width in its range then its right side must touch R . Thus we have either an extreme solution or a grid-lock solution.

To find a grid-lock solution we use procedure using-four-hyperbolas described at the end of this section.

Finally, we need to describe how to find extreme solutions. Once one of the four rectangles is fixed, it is easy to see how to fix the remaining ones. For example,

suppose we fix U to the maximum height in its range. Then we should fix T of minimum height disjoint from U , and S of maximum height disjoint from U . This leaves as much room as possible for R , which we now fix to have maximum height disjoint from T .

4.2.3.1 Procedure using-four-hyperbolas

We search for a good realization S, T, R, U for $\mathcal{S}, \mathcal{T}, \mathcal{R}, \mathcal{U}$ where:

- The right side of S and the left side of R have the same x -coordinate.
- The right side of U and the left side of T have the same x -coordinate.
- The top side of U and the bottom side of S have the same y -coordinate.
- The top side of T and the bottom side of R have the same y -coordinate.

Let t be the x -coordinate of the left side of R and assume u is in the origin. Let a_u, a_s, a_r, a_t be the areas of elastic rectangles $\mathcal{U}, \mathcal{S}, \mathcal{R}$, and $\mathcal{T}$ respectively. Then, the coordinates for:

- the right-bottom corner of S are: $(t, y(s) - \frac{a_s}{t})$.
- the right-top corner of U are: $(\frac{a_u}{y(s)-a_s/t}, y(s) - \frac{a_s}{t})$.
- the left-top corner of T are: $(\frac{a_u}{(y(s)-a_s/t}, \frac{a_t(y(s)t-a_s)}{x(t)(y(s)t-a_s)-a_u t})$.
- the left-bottom corner of R are: $(t, y(r) - \frac{a_r}{x(r)-t})$

Hence, we want to have the following equality:

$$y(r) - \frac{a_r}{x(r) - t} = \frac{a_t(y(s)t - a_s)}{x(t)(y(s)t - a_s) - a_u t} \quad (4.1)$$

From equality 4.1 we can obtain the value of t from the quadratic equation:

$$t^2(x(t)y(s)y(r)+a_u y(r)+a_t y(s))+t(x(t)y(s)y(r)x(r)-x(t)y(s)a_r+a_t a_s y(r)-a_u y(r)x(r) \\ +a_u a_r - a_t y(s)x(r) - a_t a_s) - x(t)a_s y(r)x(r) + x(t)a_s a_r + a_t a_s x(r) = 0$$

By obtaining the value of t from this quadratic expression we set U , S , T , and R to have the coordinates indicated above. We check to see if the realizations obtained for each elastic rectangle are valid realizations according to their bounds of maximum and minimum height and width. If there is a realization out of range for its elastic rectangle, then no realization is returned.

The realization obtained above may have $U \cap R \neq \emptyset$ or $T \cap S \neq \emptyset$. Note that we call this procedure when either of these two scenarios occur R is beside S and R is beside U ; or R is beside S and R is above U . Therefore, if we have either $U \cap R \neq \emptyset$ or $T \cap S \neq \emptyset$ it is because there does not exist a good realization for any of the two previous scenarios.

By construction the following claim holds.

Claim 4.6 *Procedure using-four-hyperbolas obtains a realization R, S, T, U for $\mathcal{R}, S, \mathcal{T}, \mathcal{U}$ where R touches the right side of S , S touches the top side of U , U touches the left side of T , T touches the bottom side of R .*

4.3 Discrete Two-Parallel Lines Labeling Problem

Similar to Section 3.6 the *discrete two-parallel lines labeling problem*, as its name indicates, is the two-parallel lines labeling problem of elastic rectangles with a discrete set of realizations. That is, the set of realizations of a discrete elastic rectangle

is described by a set of points rather than a hyperbola segment in the opposite corner to its anchor. However, this set of points is contained in a hyperbola segment since all realizations have the same area. Therefore, with some modifications we can use the same algorithm for the continuous case by using the hyperbola segments containing the discrete set of points.

The only part where we made use of the hyperbola segments in the two-parallel lines algorithm for the continuous case was in the criteria described in Section 4.2.3. In this section, we can see that we use the continuity of the hyperbola segments in two parts: 1) in procedure using-four-hyperbolas and, 2) when we obtain a good realization for two facing rectangles of minimum total width or minimum total height.

When we obtain a good realization of minimum total width for two facing rectangles we apply the discrete procedure explained in Section 3.6, which takes constant time.

Now, we proceed to explain how to modify the procedure using-four-hyperbolas for the discrete case. For the four cases in Section 4.2.3 the one that uses this procedure is case 4.

In case 4 we have discrete elastic rectangles $\mathcal{U}, \mathcal{R}, \mathcal{S}, \mathcal{T}$ with anchors u, r, s, t such that r, s, t are the topmost C -shape and u is below s . Here, u and t are bottom anchors and s and r are top anchors. (See Figure 4.11.) Let $\mathcal{U}', \mathcal{R}', \mathcal{S}', \mathcal{T}'$ be the continuous elastic rectangles described by the hyperbola segments that contain the set of points describing $\mathcal{U}, \mathcal{R}, \mathcal{S}, \mathcal{T}$.

Applying the procedure using-four-hyperbolas to $\mathcal{U}', \mathcal{R}', \mathcal{S}', \mathcal{T}'$ returns a good realization U', R', S', T' such that U' touches S' at the bottom, U' touches T' at the left, T' touches R' at the bottom, and R' touches S' at the right.

If U', R', S', T' are contained in the discrete elastic rectangles $\mathcal{U}, \mathcal{R}, \mathcal{S}, \mathcal{T}$, then a good realization for the discrete case was obtained.

Otherwise, we consider the set of realizations $\mathcal{S}_0$ thinner than S' in $\mathcal{S}$. By Claim 4.6 since R' maximizes the height of $\mathcal{R}$ for a realization not above of s in $\mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{U}$. Then, realizations in $\mathcal{S}$ wider than S' will not form a good realization.

We try realizations in $\mathcal{S}_0$ one by one from the widest one to the thinnest one. When trying a realization $S \in \mathcal{S}_0$ we select the tallest realization in $\mathcal{U} \in \mathcal{U}$ below S , then the widest realization $T \in \mathcal{T}$ to the right of U , and finally the tallest realization of $\mathcal{R}$ above T . We stop when we get a good realization or when the realization of $\mathcal{R}$ is above s .

By construction the following claim holds.

Claim 4.7 *The above procedure obtains a good realization R, S, T, U for elastic rectangles $\mathcal{R}, \mathcal{S}, \mathcal{T}, \mathcal{U}$ that maximizes R such that R does not lie above s .*

In comparison with the continuous case the above procedure does not take constant time any more. The time of this procedure depends of the number k of realizations we try. Thus the discrete two-parallel line algorithm takes $O(k + n)$ time, where k is the maximum number of tries of the procedure using-four-hyperbolas.

Thus, by adapting the two-parallel lines algorithm as indicated above we have an algorithm that works for the discrete case.

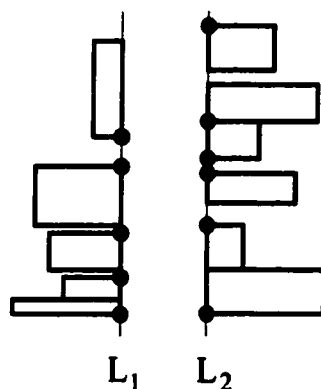


Figure 4.16: An example of a good realization for the exterior case.

4.4 One-Line Labeling Problem

A trivial variation of the two-parallel lines labeling problem is when the elastic rectangles lie not between the lines but outside as shown in Figure 4.16. Since we can solve each line independently we call this problem the *one-line labeling problem* and focus on a single line. For this variant of the problem it is clear that we can find a good realization by just taking the rectangles with minimum height in a line. It is clear that if any two of these rectangles intersect then there does not exist a good realization for the set of elastic rectangles.

However, by taking all realizations of minimum height we may have a solution with extremely wide labels. Therefore, a valid objective for this problem is to minimize the total width of the labels in each line. A more general objective is to obtain a rectangle of minimum area containing all labels on a vertical line. We proceed to explain how to solve this problem.

Note that if we have a single elastic rectangle between two anchors its height is limited by the anchors and we only have to minimize its width. We fix these rectangles to their realizations of minimum width. If we have facing rectangles, two

elastic rectangles between two anchors, the total height of the rectangles is limited as well by their anchors then we proceed to minimize their total width as indicated in Section 3.5.1. In this way we have fixed the elastic rectangles between the topmost and bottommost anchors on the vertical line. Let R be the rectangle of minimum area containing the fixed rectangles between the topmost and bottommost anchors on the vertical line. It is clear that R is of minimum area for any good realization of the elastic rectangles.

It remains to fix the elastic rectangles on the topmost and bottommost anchors in the vertical line if they lie outside R . Let $\mathcal{T}$ and $\mathcal{B}$ be the elastic rectangles of the topmost and bottommost anchors on the vertical line, respectively. If $\mathcal{T}$ has a top anchor then $\mathcal{T}$ has been fixed, and $\mathcal{B}$ has been fixed as well if it has bottom anchor. Otherwise, $\mathcal{T}$ and/or $\mathcal{B}$ remain elastic and outside of R , thus we proceed to fix them as follows.

If only $\mathcal{T}$ remains elastic we have the following cases.

1. The maximum width for $\mathcal{T}$ is smaller than the width of R . A rectangle of minimum area containing R and a realization of $\mathcal{T}$ must have the width of R . Thus we fix $\mathcal{T}$ to its realization of minimum height.
2. All realizations of $\mathcal{T}$ are wider than R then we fix $\mathcal{T}$ to its realization of minimum width.
3. There is a realization of $\mathcal{T}$ with width equal to the width of R , then we fix $\mathcal{T}$ to this realization.

We proceed in similar way if only $\mathcal{B}$ remains elastic.

If both $\mathcal{T}$ and $\mathcal{B}$ remain elastic we proceed as follows. From the realizations of minimum width for $\mathcal{T}$ and $\mathcal{B}$ we take the one of maximum width. Without loss of

generality assume elastic rectangle $\mathcal{T}$ has this realization. Then, we fix $\mathcal{T}$ as we indicated above when only $\mathcal{T}$ remained elastic. Once $\mathcal{T}$ is fixed we update R to include this fixed realization. Now we proceed to fix $\mathcal{B}$ as we explained above, when only one rectangle remains elastic.

Remember that we are using a real-RAM model of computation. Thus, fixing facing rectangles takes constant time by calculating intersection of hyperbolas as discussed in Section 3.5.1. Therefore, the algorithm takes $O(n)$ time.

An application for the one-line labeling case can be found in a recent paper by Fekete and Plaisant [FP99] in what they call “excentric labeling”. *Excentric labeling* is a labeling technique where labels are placed on two vertical lines located in opposite sides of a given circle. These labels label objects that are located inside the circle and the labels are joined to their associated object by a line. Fekete and Plaisant pointed out that information visualization systems usually lack of advanced labeling strategies. In those systems labels are often missing entirely and users have to peck at graphical objects one at the time to see the label associated to them. They suggest the use of a circle centered at the position of the cursor, and all objects lying in the interior of the circle are labeled using excentric labeling.

Chapter 5

The Rectangle Perimeter Case

In this chapter, we combine our algorithms for the two-axis and two-parallel lines labeling problems to solve the general *rectangle perimeter labeling problem*, where the elastic labels are in the interior of a rectangle with anchors lying on its perimeter. The rectangle perimeter labeling problem arises when the perimeter of the map is labeled with information about objects that lie beyond the boundary of the map, e.g. where the roads lead to. As we discussed earlier this problem is relevant in GIS where maps are displayed dynamically on a computer screen using clipping, panning, and zooming.

5.1 Definitions and Preliminaries

Given a rectangle P containing one-corner elastic rectangles in its interior, whose anchors are located in its boundary, we say that a one-corner elastic rectangle $\mathcal{E}$ is a *left-base* rectangle if its anchor lies in the left side of P with position $Q(\mathcal{E}) = \{bl\}$, or $Q(\mathcal{E}) = \{tl\}$; it is a *right-base* rectangle if its anchor lies in the right side of P

with position $Q(\mathcal{E}) = \{tr\}$ or $Q(\mathcal{E}) = \{br\}$; it is a *bottom-base* if its anchor lies in the bottom side of P with position $Q(\mathcal{E}) = \{bl\}$ or $Q(\mathcal{E}) = \{tl\}$; and it is *top-base* if its anchors lies in the top side of P with position $Q(\mathcal{E}) = \{tr\}$ or $Q(\mathcal{E}) = \{br\}$.

We say that the *base* of a bottom-base rectangle is its bottom side. Similarly, the *bases* for a top-base, right-base, and left-base rectangle are their top, right and left sides respectively.

We define the problem formally as follows.

Definition 5.1 *Given a rectangle with a set of left-base rectangles, right-base rectangles, bottom-base rectangles, and top-base rectangles in its interior, the **rectangle perimeter labeling problem** is to find a good realization for the union of left-base, right-base, bottom-base, and top-base rectangles.*

For an instance of the rectangle perimeter problem we define the *boundary rectangle* as the rectangle whose boundary contains the anchors of the set of elastic rectangles in the instance. We denote the boundary rectangle by P .

To solve the rectangle perimeter labeling problem, we show that for any instance of the problem it suffices to consider a polynomial number of decompositions of the rectangle into regions in which labels from only two sides of the rectangle compete for space. It is intuitively clear that near a corner of the map, for example, only the labels from two sides of the map are relevant. Thus, we can apply the algorithm for the two-axis labeling case presented in Chapter 3 . Similarly, we can apply the algorithm for the two-parallel lines case, presented in Chapter 4, when only the opposite sides of a boundary rectangle are considered.

5.2 Boundary Rectangle Decomposition

Consider an instance of the rectangle perimeter problem with boundary rectangle P . This section is about decomposing P into areas where labels from only two sides of P compete for space.

We aim to divide P into regions in which (see Figure 5.1) we will be able to apply the algorithms for either the two-axis or the two-parallel lines cases. Below we explain the idea behind how such decomposition is built.

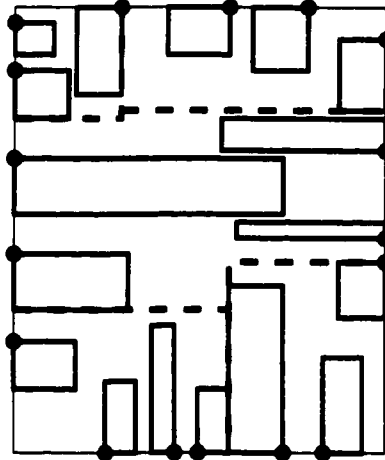


Figure 5.1: A decomposition.

Suppose we have a good realization for the elastic rectangles inside P . Let k be a corner of P and let s_1 and s_2 be the two sides of P that touch the corner k . A *corner block* at corner k is an open rectangle C contained in P with one corner at k and such that any rectangle in the good realization is either completely outside the interior of C or is completely inside C if its base lies on the sides of C lying in s_1 or s_2 . We disallow $C = P$. In Figure 5.2, we can see examples of corner blocks marked in dashed lines. Note that we allow corner blocks of height or width zero (see Figure 5.2b). However, the dashed rectangle of height zero in Figure 5.3 is not

a corner block since the bases of rectangles A and B lie on s_2 , but the rectangle does not contain rectangles A and B .

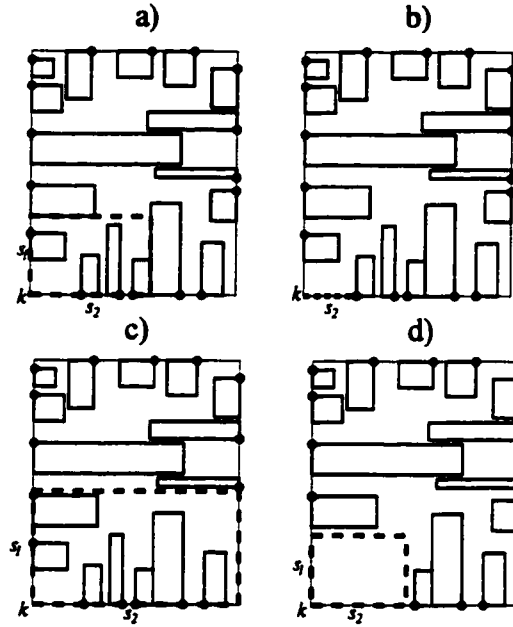


Figure 5.2: Examples of corner blocks shown in dashed bold rectangles.

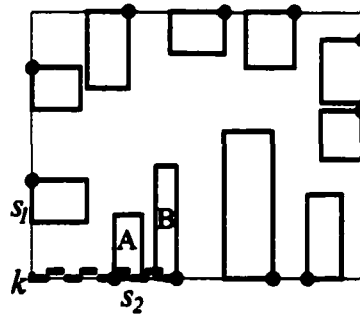


Figure 5.3: Illegal corner block.

Informally, the main idea is that inside a corner block we have an instance of the two-axis labeling problem. What about the area outside the corner blocks?

Two corner blocks *touch* if they are at adjacent corners of P and have some common boundary but disjoint interiors. Thus two touching corner blocks completely cover one side of P (see Figure 5.4). We will prove that, given any good realization of the elastic rectangles inside P , there exist four corner blocks with disjoint interiors that form two touching pairs covering two opposite sides of P . See Theorem 5.1. Such a configuration of four corner blocks is called a *corridor*, Figure 5.5 depicts a corridor. In dashed bold lines we can see the touching corner blocks covering the bottom and top sides of P .

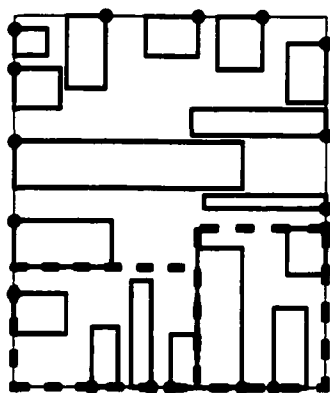


Figure 5.4: Touching corner blocks shown in dashed bold rectangles.

The area outside the four corner blocks of a corridor can only include portions of two opposite sides of P . If it includes some part of the left and/or right sides of P we call the corridor *horizontal*; if it includes some part of the top and/or bottom sides of P we call the corridor *vertical*. Thus, the area outside the four corner blocks of a corridor is either an empty region in the middle of P , or gives rise to an instance of the two-parallel lines labeling problem.

So far, the definition of corner blocks and corridors assume that we *have* a good

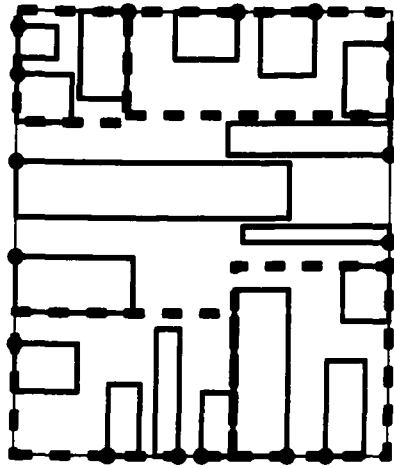


Figure 5.5: A horizontal corridor.

realization. How can we turn these ideas around to help us *find* a good realization? We would like to enumerate all the possible corridors, and for each one, solve four instances of the two-axis labeling problem for the four corner blocks, and one instance of the two-parallel lines labeling problem for the area outside the corner blocks.

Since in principle there is an infinite number of possible corridors we need to discretize them somehow. What we will do is specify the sets of labels that are to go into each of the four corner blocks. For a horizontal corridor we split the ordered list of bottom-base elastic rectangles into a left sublist for the bottom-left corner block, and a right sublist for the bottom-right corner block; we split the ordered list of top-base elastic rectangles into a left sublist for the top-left corner block and a right sublist for the top-right corner block; we split the ordered list of left-base elastic rectangles into a bottom sublist for the bottom-left corner block, a middle sublist outside all corner blocks, and a top sublist for the top-left corner block; and we split the ordered list of right-base elastic rectangles into a bottom sublist for the bottom-right corner block, a middle sublist outside all corner blocks,

and a top sublist for the top-right corner block. Such a partition will be called a *horizontal corridor partition* of the elastic rectangles. We can similarly define a *vertical corridor partition*, and define a *corridor partition* to be either one. The number of horizontal corridor partitions of the elastic rectangles is $n \cdot n \cdot \binom{n}{2} \cdot \binom{n}{2}$. Thus the number of corridor partitions is $O(n^6)$.

Theorem 5.1 justifies this approach: if there is a good realization then there is a corridor for it, and a corresponding corridor partition of the elastic rectangles.

What remains is to solve the problem: given a corridor partition of the elastic labels, is there a good realization for the elastic rectangles and a corridor for them that respects the given partition? We call this the *corridor partition realization problem*. Clearly we should use our algorithm for the two-axis labeling problem on the elastic rectangles assigned to one corner block. However, we must still deal with the fact that there is no single “best” corner block.

In Section 5.4 we describe an algorithm for the corridor partition realization problem that runs in time $O(n)$. We apply this algorithm to each of the corridor partitions. Thus, our whole algorithm runs in time $O(n^7)$.

The correctness of the algorithm comes from the existence of a corridor in a good realization (Theorem 5.1); and the guarantee that if a corridor exists for a given corridor partition then the algorithm given in Section 5.4 finds it. We prove these results in the rest of the chapter.

The remainder of this chapter is organized as follows: in the next section we prove Theorem 5.1 [the existence of a corridor]; in Section 5.4 we give an algorithm to solve the corridor partition realization problem. In Section 5.6 we discuss the discrete version of the rectangle perimeter labeling problem. The final section (5.7) is about a variation of the rectangle perimeter labeling problem where we allow

labels on the outside—but still touching the boundary—of the rectangle.

5.3 Existence of a Corridor

Given a good realization G for an instance of the rectangle perimeter labeling problem, with boundary rectangle P , in this section we prove that the instance has a corridor.

We define a *jagged horizontal line* in P as two horizontal line segments and one vertical line segment such that one horizontal line segment has one end point on the left side of P , and the other horizontal line segment has an end point in the right side of P , and the vertical line segment joins the other endpoints of the horizontal line segments. Similarly we define a *jagged vertical line*.

We say that a point in P is *free* if it is either in the interior of P but outside the interiors of the rectangles, or on the perimeter of P but outside the bases of the rectangles in P . A jagged horizontal/vertical line is *free* if all its points are free. Figure 5.6 depicts a pair of free jagged horizontal lines in a good realization.

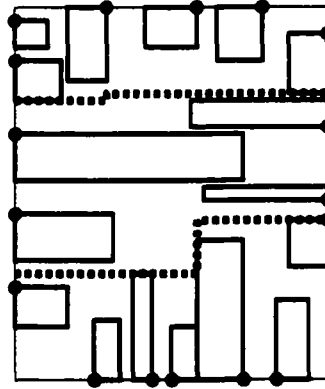


Figure 5.6: Free jagged horizontal lines.

Having the above definitions at hand we proceed to prove that G has either a free jagged horizontal line or a free jagged vertical line. This jagged line will help us to obtain a corridor.

Lemma 5.1 *Any good realization of the elastic rectangles inside P has either a free jagged horizontal line or a free jagged vertical line.*

PROOF. Let G be one of these good realizations. Now take the widest left-base rectangle L in G . We draw an auxiliary horizontal line segment l going from the left side of P to the right side such that l goes through one of the horizontal sides of L . We use the line l as an auxiliary line that helps us to obtain a free jagged line. When drawing l from left to right we have the following cases:

1. If l hits P , l is a jagged horizontal line with its vertical line empty. See Figure 5.7a.
2. If l hits a right-base rectangle R then bend l at this point to go around R to reach the right side of P . Then l is our jagged horizontal line. See Figure 5.7b.
3. If l hits a top-base rectangle T then continue the line l down. Here we have the following cases:
 - (a) If l hits the bottom side of P then we have a jagged vertical line that consists of a vertical line passing through the left side of T with its horizontal line segment empty. See Figure 5.7c.
 - (b) If l hits a bottom-base rectangle B then we bend l go around B to the bottom side of P , getting a jagged vertical line from the left side of T to the top side of B and one of the vertical sides of B . See Figure 5.7d.

- (c) If l hits a right-base rectangle then we bend l to continue on this rectangle to the right side of P getting a jagged horizontal line. See Figure 5.7e.

Note that when the line l is going down, it does not intersect any left-base rectangle since the left-base rectangles are at most as wide as L .

4. If l hits a bottom-base rectangle, we proceed in a manner similar to when it hits a top-base rectangle.

□

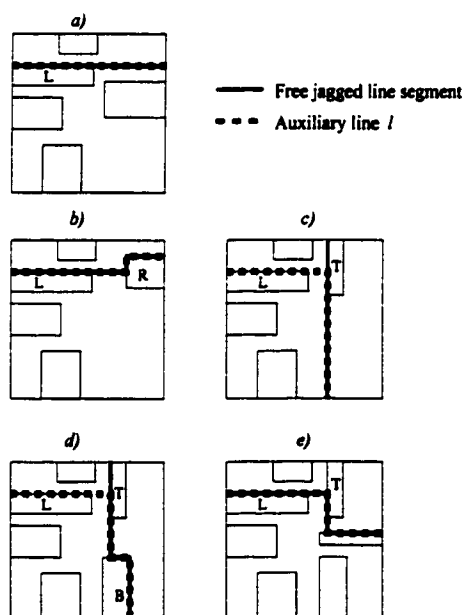


Figure 5.7: Building a free jagged line.

We say that a free jagged horizontal line h is *top-maximal* if the region in P above h is minimal by containment. In particular this means that it is not possible to make a new free jagged horizontal line by raising either of the horizontal segments in h , nor by moving the vertical segment over in such a way as to decrease the area above.

Lemma 5.2 *Let G be a good realization of the elastic rectangles inside P , and suppose there is a top-maximal jagged horizontal line h . Then there is a pair of touching corner blocks covering the top side of P with bottom sides lying on h .*

PROOF. Let v be the vertical line segment of h . See Figure 5.8. Without loss of generality assume that h has its left horizontal segment higher than its right horizontal segment. Since h is maximal we cannot obtain another free horizontal jagged line by moving v to the right and adjusting the corresponding segments to decrease the area contained above h . Thus, v is blocked by some rectangle on its right or touched a rectangle in the region between h and the top side of P . If v is blocked by a right-base rectangle R then the right horizontal segment of h touches the bottom side of R (see Figure 5.8a) and we can move to the top side of R obtaining a higher jagged line and contradicting that h is top-maximal. If v is blocked by a top-base rectangle T , i.e. v touches the left side of T (see Figure 5.8b), then we have a pair of touching corner blocks as follows, a left-top corner block with its bottom side in the left horizontal line segment of h , and with its right side at v ; a right-top corner block with its bottom side in the right horizontal line segment of h and with its right side at v . Therefore, we get a pair of touching corner blocks covering the top side of P with bottom sides on h . $\square$

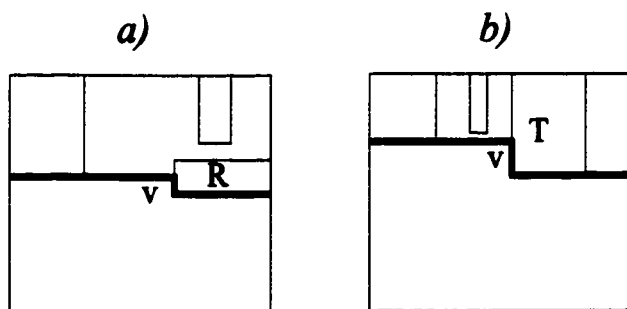


Figure 5.8: Free horizontal jagged lines.

A similar lemma holds for bottom-maximal jagged horizontal lines:

Lemma 5.3 *Let G be a good realization of the elastic rectangles inside P , and suppose there is a bottom-maximal jagged horizontal line h . Then there is a pair of touching corner blocks covering the bottom side of P with top sides lying on h .*

The proof is similar to the one of Lemma 5.2.

Using the above two lemmas we proceed to prove the following theorem.

Theorem 5.1 *Any good realization for an instance of the rectangle perimeter labeling problem has a corridor.*

PROOF. Let G be one of these good realizations. By Lemma 5.1 there is either a free jagged horizontal line or a free jagged vertical line in G . Without loss of generality assume there is a free jagged horizontal line h . Then we raise h until we get a top-maximal jagged horizontal line t . We obtain as well a bottom-maximal jagged horizontal line b by lowering h , having b below t . Then by applying Lemmas 5.2 and 5.3 we have two disjoint pairs of touching corner blocks covering the top and bottom sides of P . $\square$

When we look for a corridor we try to find both a jagged horizontal line and a jagged vertical line. This is because we do not have the guarantee that both exist. For example Figure 5.9 shows a good realization that does not have a jagged horizontal line.

Figure 5.9 shows that there does not always exist a pair of corner blocks that cover the bottom side of P . Assume that we have the good realization shown in Figure 5.9. Suppose it has a horizontal corridor. Let C_l^B and C_r^B be the left and right corner blocks covering the bottom sides of P . Then, C_l^B either contains

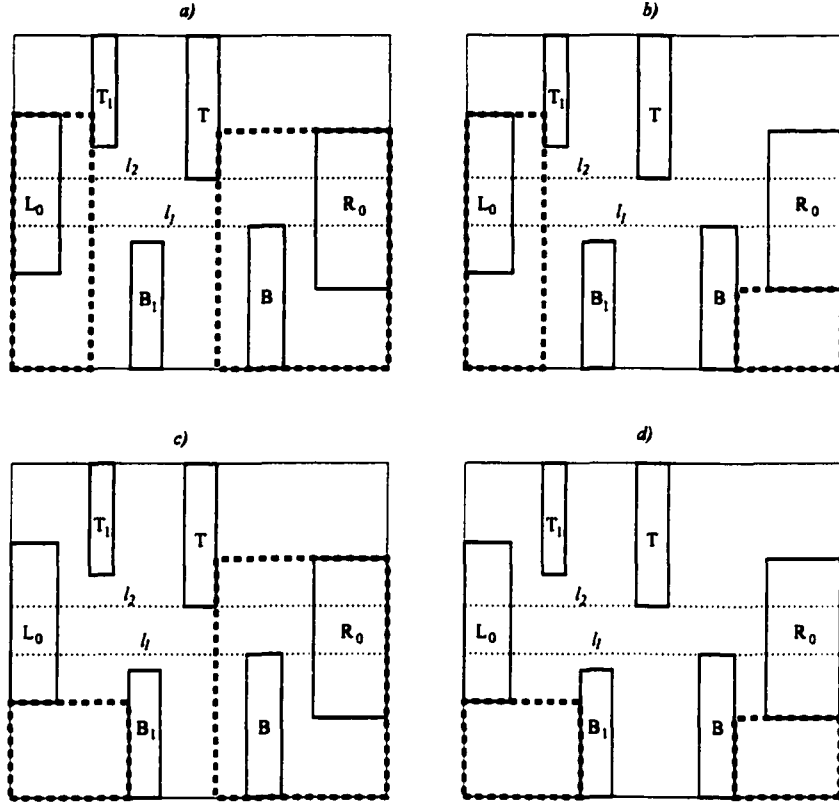


Figure 5.9: Corner blocks covering the bottom side of P cannot be built.

rectangle L_0 or does not. Suppose C_l^B contains L_0 . Thus for the instance shown in Figures 5.9a and 5.9b the right side of C_l^B must be to the left or on the left side of rectangle T_1 . Otherwise, C_l^B intersects T_1 which is not allowed for a corner block. On the other hand, if C_l^B does not contain L_0 (see Figures 5.9c and 5.9d) then its right side cannot be to the right of the left side of B_1 .

The right corner block C_r^B either contains R_0 or does not. If C_r^B contains R_0 (see Figures 5.9a and 5.9c) then its left side must be to the right or on the right side of T . If C_r^B does not contain R_0 (see Figures 5.9b and 5.9d) then its left side must be to the right or on the right side of B .

Thus in any case C_l^B and C_r^B cannot be touching corner blocks.

However, note that the instance on Figure 5.9 has a vertical corridor as shown in Figure 5.10.

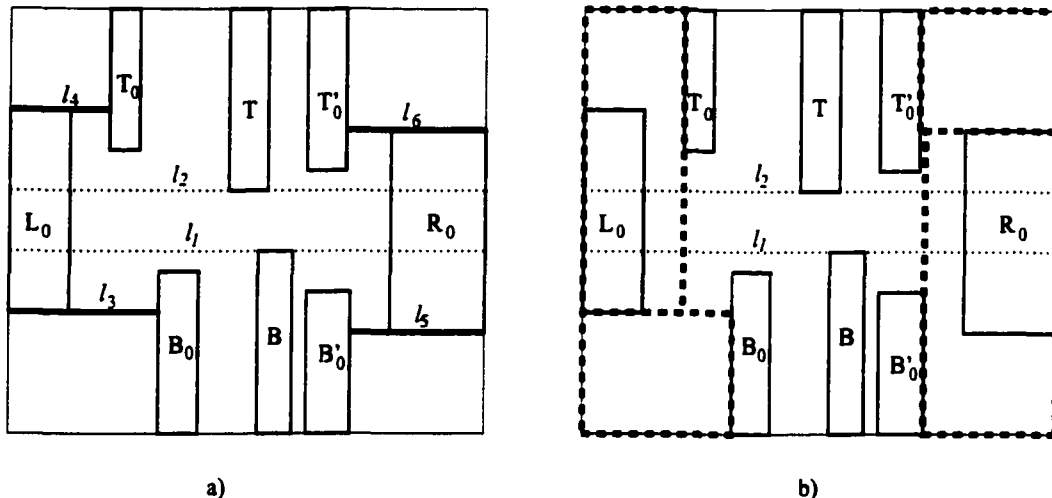


Figure 5.10: Building a Corridor when a Horizontal Corridor does not exist.

5.4 Corridor Partition Realization Problem

Here we give an algorithm to solve the corridor partition realization problem. That is, given a corridor partition for an instance of the rectangle perimeter labeling problem, we have to find if there is a good realization that forms a corridor that corresponds to the given partition.

Given an instance I of the rectangle perimeter labeling problem and a corridor partition S we proceed to give an algorithm that finds a good realization of I , if one exists, that forms the partition S .

For each corner block in S we have a list of elastic rectangles that belong to it. Thus, for each of these lists of elastic rectangles we find a good realization. We find these good realizations by applying the two-axis algorithm to each corner block.

Since we do not know yet if corner blocks indicated in S will form corner blocks in I , i.e. we do not know if they do not intersect other rectangles, we call them *blocks* to distinguish them.

If there does not exist a good realization for a list of elastic rectangles on a block, then there does not exist a corridor in I that forms the corridor partition S . Otherwise, if there is a good realization for each block what remains to do is to find a good realization for the elastic rectangles outside the blocks, and place the blocks together.

By construction the elastic rectangles outside the blocks are located on opposite sides of the boundary rectangle of I . Thus, we can apply the two-parallel lines algorithm to them.

However, we have to consider the realizations corresponding to the blocks since they may interfere with the elastic rectangles outside. We can think of blocks as if they were elastic rectangles and apply the two-parallel lines algorithm to the blocks together with the elastic rectangles outside them.

Now, we explain how we use blocks as elastic rectangles. Remember that in the two-axis algorithm we obtain two kinds of good realizations (see Chapter 3): a top-clear and a right clear realization. Remember as well that in these realizations all rectangles have been fixed but for the clear rectangle. Thus, a block preserves the elasticity of the clear rectangle.

For example, in Figure 5.11a we have a right-clear realization for elastic rectangles $\mathcal{L}_1$ and $\mathcal{B}_1$. The block of a right-clear realization is the rectangle of minimum area that contains the rectangles in the realization. Note that in this case the minimum width of the block is fixed by the right side of rectangle $\mathcal{B}_1$. Thus, the realizations of $\mathcal{L}_1$ with smaller width than the minimum width of the block are not

contained in the block. Figure 5.11b depicts the set of rectangles in $\mathcal{L}_1$ that are considered for the block. The block for this realization is shown in Figure 5.11c.

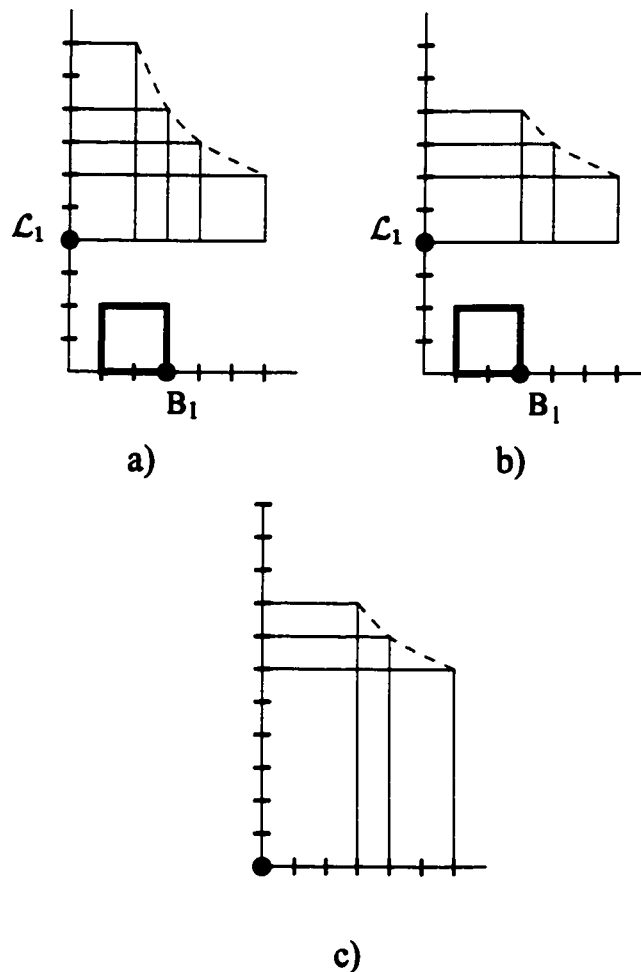


Figure 5.11: Building an elastic block.

Therefore, the block obtained in our example corresponds to the family of rectangles all having the origin as bottom-left corner and its opposite corner located in part of the hyperbola segment of $\mathcal{L}_1$.

Note that blocks do not preserve area as elastic rectangles do. In the block of Figure 5.11c if we take the minimum height allowed in the block its area is equal to

36 units, while if we take the realization of maximum height the area of the block is equal to 24 units.

Nevertheless, we are able to apply the two-parallel lines algorithm to blocks since the algorithm only needs the second-degree equation of the curve that describes a block and the position of its anchor, see section 4.2.3.

Thus, we can apply the two-parallel lines algorithm to the blocks and the elastic rectangles outside them.

Remember that for each block we could have a top-clear and a right-clear realization. Therefore, for each block we should try each of the top-clear and right-clear realizations obtained. Thus, we run the two-parallel lines algorithm at most 16 times. In total, the algorithm takes a constant factor longer than the two-parallel lines algorithm which is $O(n)$, given that the anchors are sorted.

We call the above algorithm the Corridor Partition Algorithm. Now, we proceed to prove the correctness of our algorithm.

Theorem 5.2 *Given a corridor partition for a set of elastic rectangles for an instance of the rectangle perimeter labeling problem, then there is a corridor corresponding to the partition for the set if and only if the Corridor Partition Algorithm finds it.*

PROOF. It is easy to see that if the algorithm finds a good realization, this is a corridor for the instance of the rectangle perimeter labeling problem. Thus we proceed to prove that if there exists a corridor for a given corridor partition of an instance then this algorithm finds it.

Let S be the given corridor partition for an instance I of the rectangle perimeter labeling problem. The algorithm obtains a good realization for each list of rectangles

corresponding to a corner block in S . These realizations are obtained using the two-axis algorithm. Then, blocks are obtained for each good realization returned by the two-axis algorithm. By Lemma 3.1 the two-axis algorithm obtains all minimal height-width combinations in each block. Thus, we have the guarantee that each corner block in S can be obtained from these blocks.

The algorithm then applies the two-parallel lines algorithm to the blocks and the elastic rectangles outside the blocks. Due to Theorem 4.1 we know that if a good realization exists for the blocks and the elastic rectangles outside the blocks the algorithm finds it. $\square$

5.5 Algorithm

Given an instance of the rectangle perimeter labeling problem, we find a good realization using the corridor partition realization algorithm and the two-axis algorithm as described below.

Rectangle perimeter labeling algorithm:

1. Sort the anchors along each side of P .
2. Execute the two-axis algorithm once in each corner of P , using all the elastic rectangles on each side. Recall that this algorithm preserves all subproblem solutions.
3. Generate all corridor partitions.
4. For each corridor partition of P execute the corridor partition realization algorithm.

Let n be the number of rectangles in each side of P . We proceed to explain the time that the above algorithm takes.

Sorting the anchors takes $O(n \log n)$ time.

Executing the two-axis algorithm takes $O(n^2)$ time (see Chapter 3). This algorithm uses dynamic programming, thus by running the algorithm once in each corner of P , we obtain the solution to each corner block.

Generating all corridor partitions takes $O(n^6)$ time. Executing the corridor partition algorithm takes $O(n)$ time, since we have $O(n^6)$ partitions, then the time taken by the rectangle perimeter labeling problem is $O(n^7)$.

The correctness of the algorithm follows from Theorems 5.1, 4.1, and 3.1.

The time complexity of the algorithm above can be improved somewhat, see [ILu99].

5.6 Discrete Rectangle Perimeter Labeling Problem

The discrete version of the rectangle perimeter labeling problem, as we have discussed in sections 3.6 and 4.3, is to use a set of discrete realizations for an elastic rectangle rather than an infinite set.

Note that Theorem 5.1, the existence of a corridor, holds for the discrete version of the problem. The two ingredients of the rectangle perimeter labeling algorithm that use the continuity of the elastic rectangles are the two-axis algorithm and the corridor partition realization algorithm.

In section 3.6 we gave an algorithm for the discrete version of the two-axis algorithm that takes $O(n^2)$ time, with n the number of rectangles on each axis.

The corridor partition realization problem uses the continuity of the elastic rectangles through the two-parallel lines algorithm. We presented in Section 4.3 a discrete two-parallel lines algorithm. This algorithm takes $O(n + k)$ time, with n the number of rectangles and k the number of realizations visited in an elastic rectangle.

Thus by using the discrete algorithms indicated above we can use the rectangle perimeter labeling algorithm to solve the discrete version of the rectangle perimeter labeling problem.

5.7 The Exterior Rectangle Perimeter Case

A variation of the rectangle perimeter labeling problem is where the labels are to be placed on the exterior of a boundary rectangle, *the exterior rectangle perimeter problem*. In this section we note that this problem is trivial to solve. Figure 5.12¹ depicts an example of an application of this problem. In this figure there are labels exterior to the rectangular image that contain explanations of objects located in the interior. Another application is placing labels on the x and y axes of a graph drawn in the first quadrant.

The exterior rectangle perimeter problem is an instance of the one-corner elastic labeling problem where the anchors are located on the boundary of a rectangle and their corresponding rectangles lie in the exterior of the rectangle. Figure 5.13 depicts a good realization for an instance of this problem.

The labels in the exterior rectangle perimeter problem lie in some *border* around P . In this section we give an efficient algorithm to solve the exterior rectangle

¹From: D. Powell, *Interpretation of Geological Structures Through Maps*. Wiley, 1992.

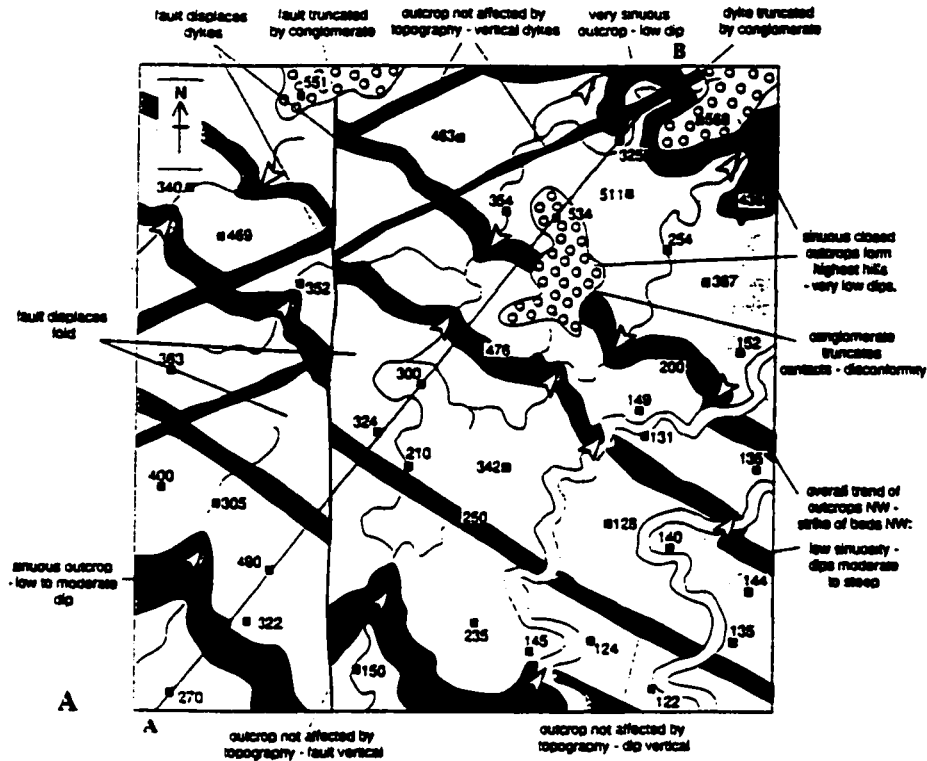


Figure 5.12: Labels in the exterior of a rectangular image.

perimeter problem while minimizing the width of the border.

We first note that the problem is trivial if we ignore the width of the border.

Let P be the rectangle containing the anchors of an instance of the exterior rectangle perimeter problem. We can find a good realization for an instance of this problem by obtaining a good realization for each side of P as follows.

For the rectangles lying on the left and right sides of P take the realizations of minimum height. It is clear that if any of those realizations intersect then there does not exist a good realization for the set of elastic rectangles. Similarly, for elastic

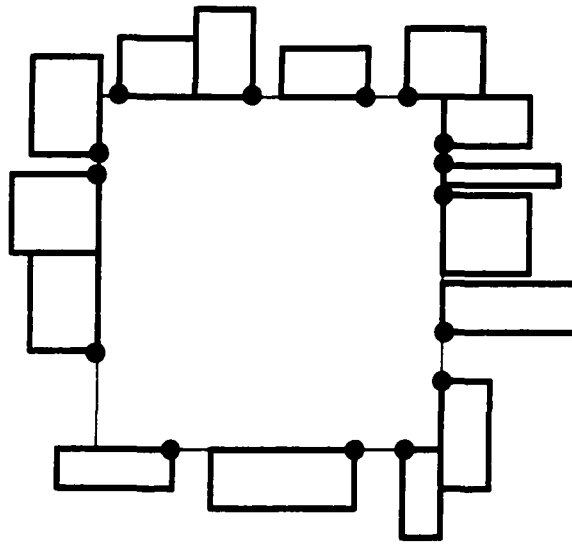


Figure 5.13: An instance of the exterior rectangle perimeter labeling problem.

rectangles at the top and bottom side of P , we take the realizations of minimum width.

Now, we just have to verify that the realization of an elastic rectangle of each side close to a corner does not intersect a realization of the other side of the corner.

Note that for any realization placed at P the side that touches P is of minimum length. Therefore, for example, if a placed realization lying at the top side of P intersects a placed realization lying at the right side of P (as shown in Figure 5.14b) then all the realizations of their elastic rectangles intersect.

In Figure 5.14, we indicate with dotted curves the hyperbolas describing the elastic rectangles. Using our criteria, in the left pane of the figure we obtain a good realization and in the right pane we have an example where a good realization does not exist.

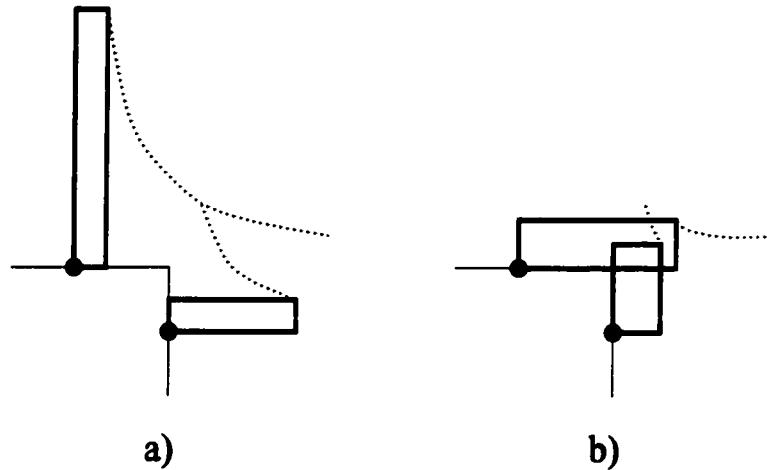


Figure 5.14: Realizations on the corners.

5.7.1 The Exterior Rectangle Labeling Problem with Minimum Border Band

In the solution given above for the exterior rectangle labeling problem we could end up with extremely thin labels. Thus, it makes sense to minimize the width of the border band containing the labels lying on P .

To solve this problem we set the labels on P as follows. In a horizontal side of P we set the labels between the leftmost and the rightmost anchors to their realizations of minimum height. This is easy to do, if we have a pair of facing rectangles we proceed as indicated in Section 3.5.1 to obtain their minimum total height realization, and if we just have a single elastic rectangle between two anchors we fixed it to its realization of minimum height. For the labels located on a vertical side of P we fix the elastic rectangles located between the topmost and the bottommost anchors to their realizations of minimum width. To fix the elastic rectangles we proceed in similar way as when the labels are on a horizontal side of P . The border band containing the fixed rectangles on the sides of P has as width the maximum

value between the height of the tallest realization fixed at the bottom and top sides of P , and the width of the widest realization fixed at the left and right sides of P .

We still have to fix the elastic rectangles lying outside the leftmost and the rightmost anchors on a horizontal side of P , and outside the topmost and the bottommost anchors on a vertical side of P . These are the elastic rectangles that we call *close-to-corner rectangles*: the ones whose realizations may extend past the corners of P . As shown in Figure 5.14, a pair of close-to-corner rectangles on adjacent sides of P may interact with each other. Therefore, we must consider them together to find a good realization.

As we can see in Figure 5.14 for each pair of close-to-corner rectangles on adjacent sides of P we have a top-clear and a right-clear realization (see Chapter 3) having in total 8 realizations to consider.

We obtain separately a good realization for each pair of close-to-corner rectangles on adjacent sides of P minimizing the width of the border band that contains them. We place these good realizations on the corners. Then the width of the border band obtained is the maximum value between the minimum widths values of the border bands obtained by realizations at the corners of P , and the width of the border band containing the rest of the realizations on the sides of P . It is clear that the border band obtained in this way is of minimum width.

Now we explain how to obtain a border band of minimum width for a pair of close-to-corner rectangles on adjacent sides of P . We explain it for the close-to-corner rectangles at the top right corner of P . For the close-to-corner rectangles of the other corners of P we proceed in similar way.

We have a top-clear and a right-clear realization to look at. Consider a top-clear realization for the pair of close-to-corner rectangles. Then, the realization of the

close-to-corner rectangle at the top side of the corner of P is fixed, and we only have to worry about the elasticity of the close-to-corner rectangle at the right side of the corner of P . Let $\mathcal{E}$ be the close-to-corner rectangle at the right side of the top right corner of P . Let a be the area of $\mathcal{E}$, and let h and w be the height and width respectively for a realization $E \in \mathcal{E}$.

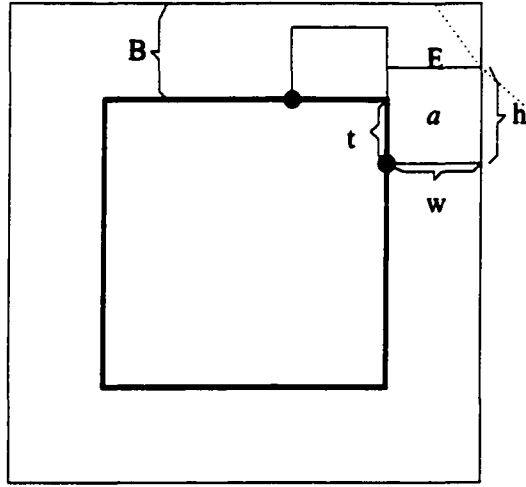


Figure 5.15: Close-to-corner rectangles.

Let B be the width of the border band, which we would like to minimize. We want a realization of $\mathcal{E}$ that is allowed to stick up into the upper right corner of the border band, but that minimizes B .

Let t be the distance from the anchor of $\mathcal{E}$ to the top side of P . We want to solve for w , h , B as follows:

$$\min B$$

$$wh = a$$

$$w \leq B$$

$$h - t \leq B$$

$$h_{\min} \leq h \leq h_{\max}$$

Claim 5.1 *The solution is either at $h = h_{\min}$ or $h = h_{\max}$, or at $w = B, h - t = B$ which means solving a quadratic equation $B(B + t) = a$.*

PROOF. We can easily see that we have obtained a realization that gives us the minimum width border band. Take a best realization E of $\mathcal{E}$. If E touches the corner of the border band R (i.e. $w = B, h - t = B$) then B has the minimum possible value. Otherwise, we must have E touching one edge of the boundary band (i.e. $w = B$ or $h - t = B$) or we could decrease B . Then we could decrease B further by trading off h and w , unless h is already at an extreme. $\square$

The time taken to fix elastic rectangles between the first and last anchor in each line determined by a side of P is linear. The time to fix a pair of close-to-corner rectangles in a corner of P is constant and we have two possible good realizations to try for them. Thus, in total we have 8 good realizations to try for all the close-to-corner rectangles. Therefore, the running time of the algorithm is $O(n)$ where n is the total number of elastic rectangles.

5.8 Rectangle Perimeter Labeling Problem with an Obstacle

A variation of the rectangle perimeter labeling problem is when there is a rectangular obstacle in the interior of P . This variant is of special interest when labels at the boundary are required not to interfere with a big graphical feature displayed in the interior of the rectangle. Having a rectangular obstacle in the middle of the

interior of P makes the labeling easier to compute since we do not have to obtain the corridor partitions of the realizations.

We draw the following auxiliary lines l_1, l_2, l_3 , and l_4 at the top, bottom, left and right sides of the rectangular obstacle inside P , respectively (see Figure 5.16). These lines partition the area between P and the rectangular obstacle into eight regions, four *corner-regions* touching the four corners of P , and four *halls* touching the four sides of the obstacle.

For the halls above and below the obstacle we fix the elastic rectangles whose anchors lie on their boundaries except for the leftmost and rightmost rectangles if they have right and left anchors respectively. We fix these elastic rectangles to their realizations of minimum total height for facing rectangles and to minimum height for the remaining rectangles.

For the halls to the left and right of the obstacle we fix the elastic rectangles whose anchors lie on their boundaries except for the topmost and bottommost rectangles if they have bottom and top anchors respectively. We fix these elastic rectangles to their realizations of minimum total width for facing rectangles and to minimum width for the remaining rectangles.

For each corner-region we find a good realization for the elastic rectangles with anchors lying on their boundaries by using the discrete two-axis algorithm. We first discard any realization of any elastic rectangle that intersects the obstacle. As in the two-axis case, for each corner-region we will have two solutions: one that is top-clear and one that is right-clear. If both of these solutions have a rectangle that intersects a hall rectangle that has been fixed, then no solution exists.

We fix the rectangles that remain elastic in the halls so that they do not intersect any fixed rectangles or the obstacle.

The one-line labeling case gives us the guarantee that we find a good realization, if one exists, for the rectangles with anchors inside the closed halls. The discrete two-axis case gives us the guarantee that we find a good realization, if one exists, for the corner-regions.

From Lemma 3.1 we have the guarantee that we have captured all minimal height-width combinations for good realizations in a corner-region. Thus if for a corner-region both top-clear and right-clear realizations intersect a rectangle in a hall then no good realization exists.

The time taken to fix the rectangles in the halls is linear with respect to the number of rectangles, and the time taken to fix the rectangles in each corner-region is quadratic with respect to the number of rectangles contained in each side of the region. When most of the rectangles lie in the halls the algorithm for this variant is faster than the algorithm for the rectangle perimeter case.

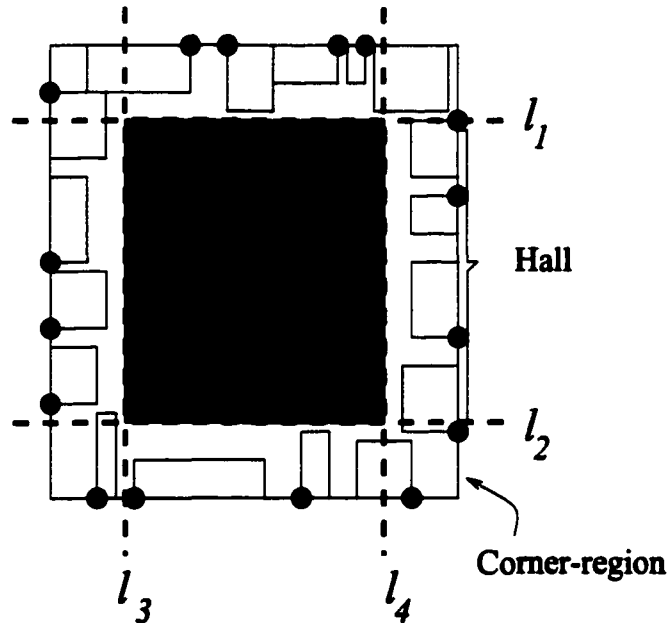


Figure 5.16: Rectangle perimeter labeling problem with an obstacle.

5.9 Conclusions

In this chapter, we addressed the elastic labeling problem when the given points lie on the boundary of a rectangular map, the so-called rectangle perimeter labeling problem. We presented a polynomial time algorithm for this problem by combining the results presented in Chapter 3 for the two-axis labeling case and in Chapter 4 for the two-parallel lines labeling case.

As we saw in Chapter 1 the general elastic labeling problem is NP-hard. Therefore, we decided to study it by adding more constraints. In this way we arrive at interesting variations that led us to the rectangle perimeter labeling problem.

As discussed in the second chapter, we can see that the rectangle perimeter labeling problem has potential applications in Geographical Information Systems (GIS).

In recent years, with the introduction of the web and GIS, dynamic maps are becoming more and more accessible to the public. We have maps that are clipped to fit their rectangular screens and we need pointers that help us to navigate them. That is, we require information about what is beyond the boundary of a map to know what place we will reach if we follow any direction beyond the boundary.

Chapter 6

Variants Using Elastic and Sliding Labels

In this chapter, we present two more variations on the point-feature label placement problem. The first problem is a variant of the one-corner labeling problem where all points must be located at the same corner on the labels. We call it the *same-corner elastic labeling problem*. The second problem is a variant of the sliding labels problem, where all points must be on the same side of their respective labels. We call it the *same-side sliding labels problem*. We present a polynomial time algorithm for each case in the following sections.

6.1 Same-Corner Elastic Labeling Problem

A variation of the one-corner labeling problem occurs when all anchors of a set of elastic rectangles are limited to the same corner of their respective rectangles: the *same-corner elastic labeling problem*. In this section, we present a sweep algorithm

to solve this problem. For convenience, we assume that the anchors are in general position, that is, no pair of anchors has the same x -coordinate or the same y -coordinate. At the end of this section we explain the case when the points are not in general position.

Consider the case where all anchors must be located at the bottom-left corners of their rectangles. The algorithm that we use is a greedy sweep algorithm. The idea is to fix each elastic rectangle during the left-to-right sweep. Remember that to fix an elastic rectangle is to select one of its realizations.

With a vertical sweep-line we go from left to right through the elastic rectangles fixing the rectangles whose anchors are to the left of the sweep-line. We fix an elastic rectangle when the sweep-line stops at its anchor. We fix it to the tallest realization that intersects none of the fixed rectangles and that contains no anchors in its interior.

The correctness of the sweep algorithm follows from the lemma below.

Lemma 6.1 *Given a good realization G of an instance of the same-corner elastic labeling problem, the sweep algorithm obtains realizations for each elastic rectangle that are greater or equal in height to the corresponding realizations in G .*

PROOF. We use induction on the number k of rectangles that have been swept by the sweep-line of the algorithm.

Let $k = 1$. Since the set of active rectangles is empty then the realization E_1 placed at elastic rectangle $\mathcal{E}_1$ is the tallest realization in $\mathcal{E}_1$ with no anchors in its interior. Thus, the lemma holds for $k = 1$.

We assume that our hypothesis is true for $k = n$ and proceed to prove it for $k = n + 1$. Let p_{n+1} be the anchor of elastic rectangle $\mathcal{E}_{n+1}$.

Since all anchors are bottom-left corners, the fixed rectangles lying completely to the left of p_{n+1} do not have conflicts with any realization of $\mathcal{E}_{n+1}$. Also the fixed rectangles with anchors below p_{n+1} do not intersect any realization of $\mathcal{E}_{n+1}$, since by hypothesis of induction those rectangles do not contain any anchor in their interior. Thus, we only have to check intersections with fixed rectangles lying above p_{n+1} .

Consider the bottommost anchor among the fixed rectangles above anchor p_{n+1} . Let p_j be this anchor. By the induction hypothesis the rectangle at p_j has the maximum height allowed in any good realization and therefore this rectangle is of minimum width for a realization $\mathcal{E}_j$ in any good realization. Then, any realization at p_j that is in a good realization G forces a realization of $\mathcal{E}_{n+1}$ to be below the y -coordinate of p_j . Therefore, the tallest realization at p_{n+1} with no anchors in its interior and below p_j is the tallest possible realization of $\mathcal{E}'_{n+1}$ in any good realization. $\square$

Now, we proceed to explain the details of the algorithm. We call the rectangles that intersect the sweep-line *active rectangles* and the rectangle that has the sweep-line at its anchor *current*.

We keep an up-to-date list of all active rectangles by adding a rectangle to the list when the sweep-line is at its anchor, and by deleting a rectangle when the sweep-line is at its right side.

The stop points of the sweep-line are the x -coordinates of the anchors—which we know from the outset—and the x -coordinates of the right sides of fixed rectangles—which we discover as we fix rectangles. To obtain the order of the stop points of the sweep-line, we use a heap. At the beginning we build a heap of the x -coordinates of the anchors, and we insert into the heap the right side of a rectangle when it is fixed.

We must still specify how the current elastic rectangle should be fixed, i.e. how we find its tallest realization that intersects none of the fixed rectangles and contains no anchors in its interior.

We begin by finding the tallest realization that intersects none of the fixed rectangles. Let c be the anchor of the current elastic rectangle. The elastic rectangles that may intersect a realization of the current rectangle are active rectangles above c . This is because the realizations with anchors to the left of the sweep-line do not contain any anchors in their interiors. Since the active rectangles are pairwise disjoint, then from the active rectangles above c we only need to obtain the active rectangle immediately above the anchor c (see Figure 6.1). To obtain the active rectangle immediately above c we maintain a balanced binary search tree with the anchors of the active rectangles. After obtaining the active rectangle R immediately above c , the tallest realization at c (if it exists) that intersects none of the fixed rectangles is the one whose top side has y -coordinate equal to the y -coordinate of the anchor of R . Note that any realization with smaller height is also disjoint from the fixed rectangles. We thus have a new upper bound on the height of a realization of the current rectangle.

We must now find a realization below this upper bound that contains no anchors in its interior. To find the tallest realization of an elastic rectangle that contains no anchors in its interior we do the following preprocessing of the anchors. Let the *range* be the area of the union of the realizations of an elastic rectangle (the area shown in Figure 2.10). Remember that a realization is an open rectangle, so the hyperbola described by them is not included in the range. For a given elastic rectangle we can obtain a list of anchors in its range by sorting the anchors by their x -coordinates. Then, we go through the sorted list and discard the anchors that are outside the range of the elastic rectangles. Once we have a list of anchors

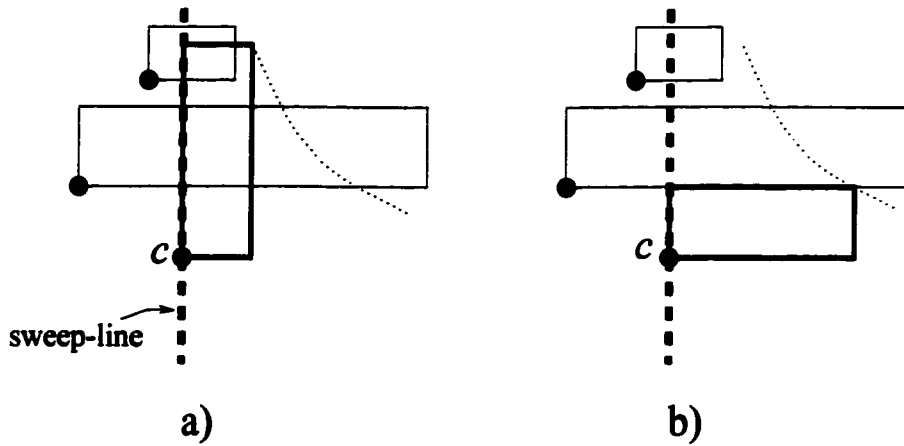


Figure 6.1: Current rectangle intersecting active rectangles.

inside the range of each elastic rectangle the following procedure obtains the tallest realization of the current elastic rectangle with no anchors in its interior.

No-anchor-inside algorithm:

1. Initialize the current realization to the tallest realization of the current elastic rectangle $\mathcal{E}$ that intersects none of the fixed rectangles.
2. Set S to be the list of anchors located in the range of the elastic rectangle sorted by their x -coordinates.
3. Go through S as follows:
 - (a) Take the next anchor (x, y) in S with y -coordinate below the top side of the current realization.
 - (b) If anchor (x, y) is outside the current realization then do nothing.
 - (c) Otherwise update the current realization to have height equal $y - p(\mathcal{E})$.

If the current realization gets shorter than the realization of minimum

height for the current elastic rectangle stop and report that no solution exists; otherwise repeat step 3.

Lemma 6.2 *Given an elastic rectangle $\mathcal{E}$ in an instance of the same-corner elastic labeling problem the no-anchor-inside algorithm returns the tallest realization in $\mathcal{E}$ without anchors .*

PROOF. Let $E \in \mathcal{E}$ be the realization returned by the algorithm. Remember that S is the sorted list of anchors that are in the range of $\mathcal{E}$. If E has maximum height then the first point in S is to the right of E , and thus all the anchors in S are to the right of E .

Suppose that E has height $h_k = y_k - p(\mathcal{E})$ which is less than the maximum height. Then the algorithm has considered a sequence I of k anchors in the range of $\mathcal{E}$ such that realization $E_i \in \mathcal{E}$ with anchor (x_i, y_i) at its top side contains anchor (x_{i+1}, y_{i+1}) , with $x_i < x_{i+1}$, $0 \leq i \leq k$, $y_0 = H$, where H is the maximum height allowed for $\mathcal{E}$. Since the shorter a realization is the wider it becomes, if a realization $E' \in \mathcal{E}$ with height h contains an anchor (x', y') then all realizations of $\mathcal{E}$ with height greater than $y' - p(\mathcal{E})$ and smaller than h still contain (x', y') . Then, all realizations with height h greater than $y_{i+1} - p(\mathcal{E})$ and smaller than or equal to $y_i - p(\mathcal{E})$ (i.e. $h \in (y_{i+1} - p(\mathcal{E}), y_i - p(\mathcal{E}))$) contain the anchor (x_{i+1}, y_{i+1}) . This means there are no valid realizations with top sides in $\cup_{i=0}^k (y_{i+1}, y_i] = (y_k, y_0]$. Then a realization containing no anchors should have height smaller or equal to $y_k - p(\mathcal{E})$. Now, since the algorithm stops at realization E , whose anchor is the k -th in the sequence I of anchors, then there are no anchors inside it. Therefore, E is the tallest realization containing no anchors. $\square$

The preprocessing for finding the anchors inside the range of each elastic rectangle can be done in $O(n^2)$ time, where n is the number of anchors.

The time taken by the sweep algorithm to find a good realization, if one exists is as follows. To sort the stop points of the sweep-line we build a heap with the x -coordinates of the stop points reporting the minimum of these values. First, we build this heap containing the x -coordinates of the anchors which can be done in linear time. Then, inserting in the heap the stop points of the sweep-line indicated by the x -coordinate of the right sides of the fixed rectangles takes $O(\log n)$ time. Obtaining the active rectangle immediately above the anchor of the current rectangle takes $O(\log k)$ time, where k is the number of active rectangles at this step in the algorithm, $k \leq n$. Checking if this active rectangle intersects the current rectangle takes constant time, by checking if their vertical sides intersect. To obtain the tallest realization for a given elastic rectangle without anchors in its interior takes $O(k')$ time, where k' is the number of anchors in the range of the elastic rectangle, $k' \leq n$. Therefore, the time taken by the sweep algorithm to find a good realization, if one exists, is $O(n^2)$ time.

We believe that a faster algorithm may be obtained by computing a realization of maximum height with no anchors in its interior in $o(n)$ time. This may be possible by the introduction of an appropriate partial order on the anchors contained inside a range of an elastic rectangle or by finding a bound on the number of times the algorithm may examine a linear number of anchors.

We can remove the restriction of anchors being in general position and the algorithm still works. Take two points p and q with the same y -coordinate, with p to the left of q . When the sweep-line stops at point q the algorithm has fixed the rectangle at point p . Either the rectangle at p is to the left of q or it contains q on its bottom side. We assume that it contains q on its bottom side. From Lemma 6.1, the rectangle at p is the tallest one for any good realization of the whole set of rectangles. Therefore, the rectangle at p is the thinnest possible one for a good

realization. Thus, the options to avoid an intersection with a rectangle at q is to choose a realization for the elastic rectangle at p below q or a realization for the elastic rectangle in q below p , which is not possible. Therefore, this instance of the problem does not have a good realization and the algorithm reports that a good realization does not exist. It is clear that for anchors having the same x -coordinate that we must fix their rectangles in order from top to bottom.

6.2 Same-Side Sliding Labels Problem

We shall prove in Chapter 8 that the sliding labels problem is NP-hard. In this section we study a variation of the problem that can be solved in polynomial time. Remember that for the sliding labels problem each label is a rectangle of fixed height and width. For the definitions and terms used for the sliding labels problem refer to Chapter 2 .

The variation consists of having all anchors located on the same side of the labels, that is, either all anchors are on the left or all anchors are on the right sides of the labels. The problem is to find the position of the labels on the anchor such that the labels do not intersect each other. We call this problem the *same-side sliding labels problem*.

We assume that the anchors are in general position and study the case when anchors are on the left side of their labels. The case when the points are not in general position is explained at the end of this section. Our algorithm is a greedy sweep algorithm going from top to bottom. Given an instance of the same-side sliding labels problem we visit its anchors in top to bottom order. We fix a rectangle when we visit its anchor. Thus the anchors above the visited anchor have

their rectangles fixed. We fix a rectangle to the highest realization that intersects none of the fixed rectangles.

Lemma 6.3 *Given a good realization G for an instance of the same-side sliding labels problem, the algorithm obtains realizations for each label that are positioned higher than or equal to their corresponding realizations in G .*

PROOF. We use induction on the number of anchors k in G visited by the algorithm. Let $k = 1$ and p_1 be the first visited anchor in G . Thus, p_1 is the topmost anchor in G . Since there are no anchors above p_1 the algorithm fixes the rectangle at p_1 to its highest possible position for a realization in p_1 . Since the anchors are in general position we do not have anchors with the same y -coordinate, at any given step there is no ambiguity about which anchor to select. This is not the case if the points are not in general position as we show at the end of this section.

Assume that the lemma is true for $k = n$, and we proceed to prove it for $k = n + 1$. Let p_{n+1} be the $(n + 1)$ -st visited anchor from top to bottom. The algorithm fixes the rectangle at p_{n+1} to its highest possible position not intersecting the fixed rectangles above p_{n+1} . By the induction hypothesis we know that the rectangles placed at the first top n anchors are placed as high as possible. Thus, we cannot move up any of the rectangles above p_{n+1} , as to be able to move the rectangle at p_{n+1} to a higher position. $\square$

We find the highest realization for the visited anchor c as follows. Place the highest possible realization at c , i.e. the one having c as bottom-left corner. Then check if the rectangles with anchors above c intersect the realization placed at c . Take the lowest bottom side of the rectangles intersected, and move the realization at c down to have its top side at the same y -coordinate of this bottom side. If it is

not possible to move the realization at c down in this way then no good realization exists.

It takes $O(k - 1)$ time to check for intersections with the rectangle placed at anchor k and the $(k - 1)$ fixed rectangles with anchors above k . Therefore, the running time of the algorithm is the time for sorting the anchors, $O(n \log n)$ with n being the number of anchors, plus the time for checking the intersections with the fixed rectangles, $O(n^2)$, for a total time of $O(n^2)$.

The algorithm does not work if the anchors are not in general position. Figure 6.2 depicts an example where the algorithm does not work when we remove the restriction of anchors being in general position. In Figure 6.2a, we can see that the algorithm does not obtain a good realization for the given set of rectangles, even though there exists a good realization for this set of rectangles as presented in Figure 6.2b. In Figure 6.2a, the algorithm may select the rectangle at p_1 as high as possible, that is, having p_1 at the bottom corner. Then the rectangle at p_2 is forced to have p_2 at the top corner. In consequence, it forces the rectangle at p_3 to have p_3 at the bottom corner and therefore leaves no available option to avoid with anchor p_4 in the interior of the rectangle at p_3 . The algorithm failed, because it attempted to place p_1 at its highest position.

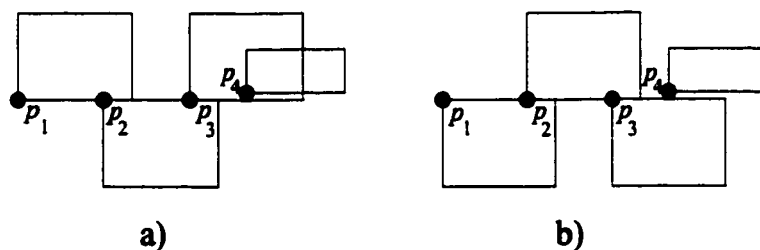


Figure 6.2: An instance of the same-side sliding labels problem with anchors not in general position.

Chapter 7

Approximation Algorithms for Labeling

The point-feature label placement problem is known to be NP-complete. Several variations of this problem using sliding labels or elastic labels result in problems that are NP-hard. Hence, a good alternative is to look for approximation algorithms for these problems. In this chapter, we present some approximation results for variations of the point-feature label placement problem. Section 7.1 is a survey of what is known. Sections 7.2 and 7.3 contain our approximation algorithms for variations of the point-feature label placement problem.

7.1 Introduction

Approximation algorithms have been developed in response to the impossibility, provided $P \neq NP$, of solving important optimization problems efficiently. A good reference on approximation algorithms is the book of Hochbaum [Hoc97]. Bern

and Eppstein [BEp97] provide a good survey on NP-hard geometric problems. One of these problems is the point-feature label placement problem. Although Kučera et al. [KMP93] gave two exact sub-exponential algorithms for the decision problem, these algorithms are not practical for common real world applications. Thus, approximation algorithms for the problem with better running times have been studied. Approximation algorithms for the point-feature label placement problem tend to follow one of two approaches:

1. find the maximum size or scaling of the labels that can be used, having all points labeled with the same size labels, or
2. label a maximum size subset of points.

One motivation for the first approach is to find the maximum font size, and for the second to use a fixed font size for as many labels as possible. In this chapter we take the second approach. First, we give some definitions, then an overview of the work that has been done, and finally we present two approximation algorithms for variants of the point-feature label placement problem.

We are working on a maximization problem for which the potential solution has a positive value. Let A be an algorithm for an optimization problem. When A is applied to an instance I of a problem we denote the produced result as $A(I)$.

Definition 7.1 *A polynomial algorithm A for some maximization problem is said to be a δ -**approximation** algorithm if for every problem instance I with an optimal solution value $OPT(I)$ such that $OPT(I) \leq \delta A(I)$.*

Note that in all cases $\delta \geq 1$. Sometimes in these approximation algorithms we will be willing to invest more running time in order to get a better approximation. Such a trade-off is common in several approximation schemes.

Definition 7.2 A family $\{A_\epsilon\}$ of approximation algorithms for a maximization problem P , is called a **polynomial approximation scheme** if for each $\epsilon > 0$ algorithm A_ϵ is a $(1 + \epsilon)$ -approximation algorithm that runs in time $O(n^c)$, where c is a constant that may depend on ϵ .

We present first a brief summary of the work that has been done on approximation algorithms to find the maximum size or scaling of the labels that can be used to label all points with the same size labels. Then, in the rest of the chapter we concentrate on the problem of labeling as many points as possible.

For the point-feature label placement problem where a label can be positioned with any of its four corners at the associated point, i.e. four candidate positions per label, the following results are known. Given a set of n points in the plane, we wish to find the maximum possible square label for which all points can be labeled with pairwise disjoint labels and the points are in any of the corners of their labels. Formann and Wagner in [FWa91] gave an $O(n \log n)$ time algorithm that guarantees labels of side length of at least half the optimum side length. They also show that no better approximation is possible unless $P = NP$. Wagner [Wag94] showed that an approximation algorithm that achieves such a bound must take $\Omega(n \log n)$ time.

The algorithm presented by Formann and Wagner starts with very small square labels, and then proceeds to increase their side lengths. At each step a candidate position (at most four) of a label may be removed if doubling the side length of the label causes some point to lie inside the label. They continue with this process until the problem can be solved for the remaining candidate position labels by running a 2-SAT algorithm.

Doddi et al. [DMM97] generalized this problem by giving approximation algorithms that allow a point to be located in any part on the boundary of a label, and

labels can have different orientation. More specifically:

1. Given a set of points find the maximum side length such that all points are labeled with pairwise disjoint squares having this side length, with points in any part of the boundary of their labels, and the labels having any orientation. For this problem, they gave an $8\sqrt{2}/\cos(\pi/5)$ -approximation algorithm that runs in $O(n \log n)$ time.
2. Given a set of points find the maximum radius such that all points are labeled with pairwise disjoint circles having this radius and with the points located in any part of the boundary of their labels. For this problem, they gave a $4(2 + \sqrt{3})$ -approximation algorithm that runs in $O(n \log n)$ time.
3. They proved the existence of a constant-approximation algorithm for each type of regular polygon labels for labeling all points of a given set with same size labels and their associated point located at the boundary of its label.

An equivalent scaling problem is to label rectilinear segments with rectangular labels as follows: Given n rectilinear segments the problem is to find the maximum width w , such that there exists a placement for n non-overlapping rectangular labels, each attached to a distinct line segment, of width w and of length the same as the line segment. We say that for an horizontal segment the width of its label is defined by the length of its vertical sides, and for a vertical segment the width of its label is defined by the length of its horizontal sides. Poon et al. [PZC98] gave an $O(kn^2 \log n)$ time algorithm to place a label per segment where $k < n$ is the number of candidate positions for each segment.

This completes the summary of some approximation results for interesting variations of the point-feature labeling problem. We now focus on the following problem.

Definition 7.3 *Given a set of points in the plane each associated with a rectangular label, where a label has a discrete number of candidate positions with its associated point located at the boundary, the **maximum sized subset labeling problem** is to find the maximum number of labels that can be placed at their associated point such that they are pairwise disjoint.*

This time, we consider the rectangular labels to be topologically closed. A point must be in the boundary of its associated label. Thus, any two candidate positions of a label at its associated point intersect. Therefore the maximum sized subset labeling problem is equivalent to finding the maximum independent set of the intersection graph of rectangles. Obtaining a maximum independent set for intersection graphs of rectangles is known to be NP-hard [FPT81, IAs83].

It is known that no polynomial-time $\Omega(n^{1/4})$ -approximation algorithm exists for maximum independent sets for arbitrary graphs [BSu94] (unless $P=NP$), but no such bound is known for intersection graphs of rectangles. For a set of unit squares, Hunt et al. [HMR95] gave a polynomial time approximation scheme to find the largest size subset of unit squares that do not intersect. Agarwal et al. [AKS98] extended this result for rectangles of unit height and arbitrary width. They gave a 2-approximation algorithm for the maximum pairwise disjoint subset in a set of n unit-height axis-parallel rectangles, which takes $O(n \log n)$ time. As well they provided an $(1 + \frac{1}{k})$ -approximation algorithm whose running time is $O(n \log n + n^{2k-1})$ for any integer $k \geq 1$. For this algorithm they combine a dynamic programming algorithm with the shifting technique of Hochbaum and Maass [HMa85].

The problem of finding a constant-approximation algorithm for the maximum independent set of arbitrary axis-parallel rectangles remains open. Agarwal et al. [AKS98] provided an $O(\log n)$ -approximation algorithm for this problem that runs in $O(n \log n)$

time.

The results presented above are for the case when we have a finite number of candidate positions to place a label, as is the case for the point-feature variation of the map labeling problem. Van Kreveld et al. [KSW98] study the case of the sliding version of the map labeling problem, where there is a continuous range of candidate positions. They consider closed rectangles as well, that is labels are not allowed to touch. They gave a 2-approximation algorithm that takes $O(n \log n)$ time for rectangles of the same height. They also proved that for any constant $\epsilon > 0$, a polynomial time $(1 + \epsilon)$ -approximation algorithm exists for the sliding version of the maximum sized subset labeling problem. Their algorithm can be adapted to labels of varying heights, but the approximation factors do not hold any more. They also provide results that suggested that the sliding model is 10-15% better than the model used for the point-feature labeling model with respect to the number of labels placed.

It remains open to find a constant-approximation algorithm for the maximum sized subset labeling problem, if one exists, for arbitrary height rectangles. In the next sections, we present approximation algorithms for two variations of this problem.

7.2 Squares of Different Sizes

As we discussed in the previous section, there is an $O(\log n)$ -approximation algorithm [AKS98] for finding the maximum independent subset of arbitrary rectangles. However, the existence of a constant-approximation algorithm for this case is an open problem. Constant approximation algorithms running in polynomial time have been developed for the same size squares [HMR95], and for rectangles of same

height [AKS98]. In this section, we provide a constant approximation algorithm for squares of different sizes.

Assume we have a set of square labels of different sizes in the plane and the goal is to find the maximum number of squares that do not intersect.

The idea of the algorithm is to classify the squares by the length of their sides. For each side length we make a subset containing all the squares of this size. Then, for each subset of squares we find a maximal subset of squares that are pairwise disjoint. The algorithm proceeds as follows.

Different-size-square algorithm:

1. Let S to be the set of squares.
2. For each square in S obtain a list of squares that intersect it.
3. Sort the side lengths of the squares in increasing order.
4. Let S_i be the subset of squares of S with side length s_i .
5. For each side length s_i in increasing order do the following:
 - (a) Select a square $r \in S_i$ and delete it from S_i , including $i = j$.
 - (b) Delete all the squares that intersect r from all subsets S_j .
 - (c) Repeat this steps a and b until S_i is empty.

Lemma 7.1 *Given a set of squares S , with a maximum independent subset M and a subset D obtained from S by applying algorithm different-size-square, then for each square $s \in M$ either s is in D or s intersects a square in D of equal or smaller size.*

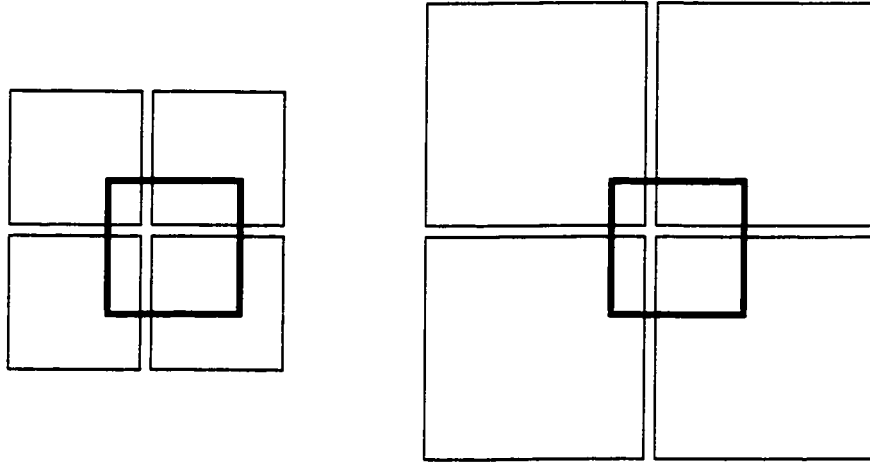


Figure 7.1: A maximal independent set for squares (indicated in bold squares).

PROOF. Let s be a square in M . If s is not in D then let S_i be the subset of S containing all squares of size equal to the size of s . Now, consider the moment that subset S_i is considered in step 5 by algorithm *different-square-size*. Then, either square s has been deleted from S_i in a previous step or not. At this point the algorithm has considered squares of smaller size than the size of s . If s has been deleted from S_i , then s was intersected by square in D of smaller size. Otherwise, s will be included in D or will be deleted from S_i because intersects a square in D of the same size. $\square$

Lemma 7.2 *Given a set of n squares S not necessarily of the same size, the different-size-square algorithm is a 4-approximation algorithm that takes $O(n \log n)$ time for the problem of finding a maximum independent subset of S .*

PROOF. Let OPT be an optimal solution for the maximum independent set of S , and let D be the subset of S obtained by applying the *different-size-square* algorithm to S . We know that $|D| \leq |OPT|$.

Let $s \in OPT$. By Lemma 7.1 we have that s is intersected by at least one square in D of the same or smaller size. Thus, we associate a square of OPT to a square of D as follows. If a square $s \in S$ is both in OPT and in D we associate s to itself. If s is in OPT but not in D , then, we associate s to any of the squares in D of the same or smaller size that intersect it.

It is easy to see that for a given square r there are at most four pairwise disjoint squares of the same or bigger size than r that intersect it (see Figure 7.1). Thus, a square in D has associated at most four squares of OPT . Hence, $OPT \leq 4D$.

The running time of the algorithm consists of the following. Obtaining a list of intersecting squares for all squares in the set takes $O(n \log n)$ time [PSH85]. Sorting the length sides of the squares takes $O(n \log n)$ time. Deleting all intersected rectangles takes $O(k)$ time, with $k \leq n$. Therefore, the algorithm takes $O(n \log n)$ time in total. $\square$

7.3 Rectangles of Different Heights

In this section, we propose another variant where we favor rectangles according to the importance of the feature labeled by them. This variation of the problem is particularly useful, for example, in maps, where labels represent places of different degrees of relevance. Thus, given the significance of the places to be labeled we want to label first the more important ones. That is, if we had to choose between the label of Toronto and the label of Mississauga we would prefer the label of Toronto.

To solve this problem, we provide a variation of Agarwal et al. [AKS98] approximation algorithm for rectangles of arbitrary heights. Although the variation is simple the problem that it solves is very practical.

The algorithm of Agarwal et al. [AKS98] is as follows. Let n be the number of axis-parallel rectangles given. Sort the horizontal edges of the rectangles by the y -coordinates and their vertical edges by their x -coordinates. Then, if $n \leq 2$, compute the maximum independent set in $O(1)$ time. Otherwise, proceed as follows.

1. Let x_{med} be the median x -coordinate among the abscissas of the vertical sides of the rectangles.
2. Partition the set of rectangles into three groups: R_1 , R_2 , and $R_{1,2}$, where $R_{1,2}$ contains rectangles intersecting the line l with x -coordinate equal to x_{med} , and R_1 and R_2 contains the rectangles lying to the left and right of the line, respectively.
3. Compute $I_{1,2}$ the maximum independent set of $R_{1,2}$. Recursively compute I_1 and I_2 , the approximate maximum independent sets in R_1 and R_2 , respectively.
4. If $|I_{1,2}| \geq |I_1| + |I_2|$, return $I_{1,2}$, otherwise return $I_1 \cup I_2$.

The key step of this algorithm is that for rectangles in $R_{1,2}$ the algorithm finds the maximum independent set for the intervals obtained by the intersection of the rectangles in $R_{1,2}$ with line l . Agarwal et al. [AKS98] proved that the above algorithm is a $O(\log n)$ -algorithm that takes $O(n \log n)$ time.

The variation that we propose to this algorithm is to find the maximum weighted independent set instead of the maximum independent set. We associate the relevance of a place as the weight of its label. For example, in labels of cities we can associate as weight their population. We can use the above algorithm by obtaining the maximum weighted independent set in step 3, instead of the maximum independent set; and by replacing step 4 by:

4. If $\text{weight}(I_{1,2}) \geq \text{weight}(I_1) + \text{weight}(I_2)$, return $I_{1,2}$, other wise return $I_1 \cup I_2$.

We call the modified algorithm **weighted**.

Hsiao and Tang [HTa92] gave a dynamic programming algorithm that obtains the maximum weighted independent set on an interval graph. This algorithm runs in linear time when a sorted list of the endpoints of the intervals is given.

We keep two lists of sorted points: one list for the x -coordinates of the vertical sides of the rectangles, ordered from left to right, and another list for the y -coordinates of the horizontal sides of the rectangles. We use the sorted list containing the x -coordinates of the vertical sides of the rectangles to obtain x_{med} . At the time that we obtain x_{med} we split the list into two sorted sublists, that correspond to the vertical sides of the rectangles R_1 and R_2 , respectively. We use the sorted list containing the y -coordinates of the horizontal sides of the rectangles to obtain the maximum weighted independent set of the rectangles intersected by line l . We split this list into three sorted sublists that correspond to the horizontal sides of the rectangles to the left of l , to the right of l , and for the rectangles intersected by l . We obtain these three sublists by going through the original list in order and placing each point in the corresponding sublist in the same order. We call this process **lists**. We can see that **lists** takes linear time. These sublists allow us to use the linear algorithm of Hsiao and Tang [HTa92] for obtaining the maximum weighted independent set of the rectangles intersected by l .

From the above discussion we only need to sort once the x -coordinates and y -coordinates of the vertical and horizontal sides of the rectangles. Having these sorted lists we apply the **weighted** algorithm.

Let $T(n)$ be the running time taken by the **weighted** algorithm on n given

axis-parallel rectangles. We have:

$$T(n) = \text{lists}(n) + T(|R_1|) + T(|R_2|) = O(n) + T(|R_1|) + T(|R_2|)$$

We have that $|R_1| \leq n/2$ and $|R_2| \leq n/2$, then

$$T(n) \leq cn + T(n/2) + T(n/2) = \sum_{i=1}^{\log n} cn = cn \log n$$

Therefore the total time of the weighted algorithm is $O(n \log n)$.

The weighted algorithm computes a maximum weighted independent set of size at least $B / \max(1, \log n)$, where B is the maximum weighted independent set. The proof of the approximation result for the weighted version follows from the proof of Agarwal et al. [AKS98] approximation result as follows.

The proof is by induction. We can see that the statement is true for $n \leq 2$. Suppose is true for all $m < n$. Let R be the set of rectangles and M be the maximum weighted independent set. Similarly let $M_1, M_2, M_{1,2}$ be the maximum independent sets of R_1, R_2 , and $R_{1,2}$, respectively. The weighted algorithm obtains the maximum weighted independent set $I_{1,2}$ for $R_{1,2}$. Then we have:

$$\text{weight}(I_{1,2}) = \text{weight}(M_{1,2}) \geq \text{weight}(M \cap R_{1,2})$$

By the induction hypothesis

$$\text{weight}(I_1) \geq \text{weight}(M_1) / (\log(n/2)) \geq \text{weight}(M \cap R_1) / (\log n - 1)$$

$$\text{weight}(I_2) \geq \text{weight}(M_2) / (\log(n/2)) \geq \text{weight}(M \cap R_2) / (\log n - 1)$$

Then we have

$$\text{weight}(I) = \max\{\text{weight}(I_{1,2}), \text{weight}(I_1) + \text{weight}(I_2)\}$$

$$\begin{aligned} &\geq \max \left\{ \text{weight}(M \cap R_{1,2}), \frac{\text{weight}(M \cap R_1) + \text{weight}(M \cap R_2)}{\log n - 1} \right\} \\ &\geq \max \left\{ \text{weight}(M \cap R_{1,2}), \frac{\text{weight}(M) - \text{weight}(M \cap R_{1,2})}{\log n - 1} \right\} \end{aligned}$$

If $\text{weight}(M \cap R_{1,2}) \geq \text{weight}(M)/\log n$ then the induction step is proved.

Otherwise:

$$\frac{\text{weight}(M) - \text{weight}(M \cap R_{1,2})}{\log n - 1} \geq \frac{\text{weight}(M) - \text{weight}(M)/\log n}{\log n - 1} = \frac{\text{weight}(M)}{\log n}$$

Therefore, the weighted algorithm remains an $O(\log n)$ -approximation algorithm.

Chapter 8

NP-hardness

It is known that the point-feature label placement problem is NP-complete [FWa91, KIm88, KRa92, MSh91]. In this chapter, we show the NP-hardness of two different formulations of the point-feature label placement problem: the *one-corner elastic labeling problem* and the *sliding labels problem*. We also show that the point-feature label placement problem remains NP-complete when only three positions of a label are considered. We call this the *three-position point-feature labeling problem*.

8.1 One-Corner Elastic Labeling Problem

As we discussed in Chapter 2, the elastic labeling problem is NP-hard, since it generalizes the point-feature label placement problem in that we are free to choose not only which corner of the label to place at the point, but also the height and width of each rectangular label as long as we achieve the specified area. In this section, we show that even when we fix the corner of each elastic rectangle leaving only the choice of height and width the problem remains NP-hard. This is called

the *one-corner elastic labeling problem*.

Theorem 8.1 *The one-corner elastic labeling problem is NP-hard.*

PROOF. Note that we are not claiming NP-completeness, because it is not clear that the problem is in NP. For a problem to be in NP we must have a polynomial size certificate for positive instances. The natural certificate for a labeling problem is a good realization. Given a candidate realization one can check if it is a valid certificate, i.e. a good realization, by checking for intersections. However, the problem is that a realization of a rectangle may have height or width that are real numbers with infinite representation, which makes the certificate too big.

We will reduce 3-SAT to the one-corner elastic labeling problem, following the idea of the proof given by Formann and Wagner [FWa91] for the NP-completeness of the point-feature label placement problem.

Given an instance of 3-SAT, we will construct an instance of the one-corner elastic labeling problem, with gadgets for the variables and clauses, and with “pipes” joining them, in such a way that the 3-SAT formula is satisfiable if and only if the set of elastic rectangles has a good realization.

The basic building block is a *toggle*, as shown in Figure 8.1. Point q is a dummy point, with an associated 0×0 rectangle. The elastic rectangle associated with point p has area t , height range $[1, t]$, and width range $[1, t]$. The presence of point q forces us to use one of two rectangles for point p , either $1 \times t$, or $t \times 1$.

Toggles can be joined to form horizontal and vertical pipes as shown in Figure 8.2. We consider these pipes to be directed, from right to left, and from bottom to top, respectively. Observe that if the first toggle in a pipe is realized (or “set”)

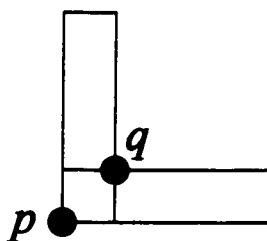


Figure 8.1: Basic building block.

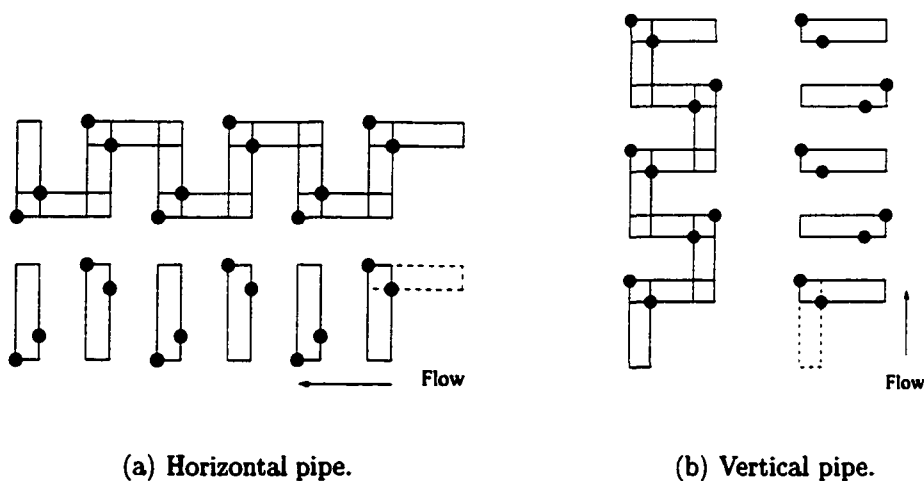


Figure 8.2: Pipes.

perpendicular to the pipe, then all the toggles in the pipe must also be set perpendicular to the pipe. In this case we say that there is “flow” in the pipe. (Note that we can use larger values of t to “stretch” some of the toggles in a pipe to align things nicely.)

For each variable x , construct two overlapping toggles connected to two horizontal pipes, labeled p_x and $p_{\bar{x}}$, as shown in Figure 8.3. Observe that there can be flow in at most one of the pipes. Setting variable x True [False] will correspond to having flow in pipe p_x [$p_{\bar{x}}$, respectively] in order to have a good realization for the whole set.

For each clause, construct an elastic rectangle with three realizations, connected to three vertical pipes as shown in Figure 8.4. The three vertical pipes will correspond to the three literals in the clause. Observe that there must be flow in at least one of the three vertical pipes.

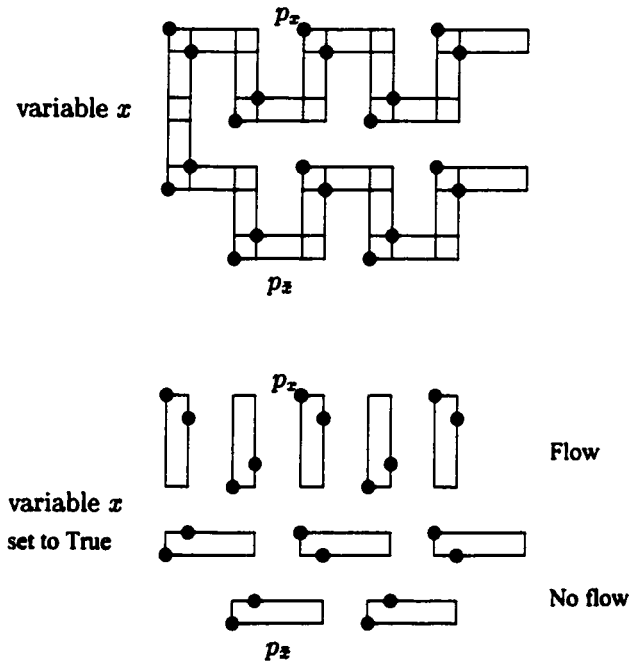


Figure 8.3: Variable setting.

The over-all structure of the instance of the one-corner labeling problem is shown in Figure 8.5. The variables are lined up on the left, with two horizontal pipes for each one. The clauses are lined up on the bottom, with three vertical pipes for each one. Each vertical pipe corresponds to some literal in some clause, and must be joined with the horizontal pipe for that literal. The connection is shown in Figure 8.6. Observe that if there is flow in the vertical pipe, then there must be flow in the horizontal pipe.

The one thing we still need is a crossing gadget that allows a vertical pipe to cross a horizontal pipe without disturbing flow in either one. Suppose we have such a crossing gadget. Let us prove that the 3-SAT formula is satisfiable if and only if the set of elastic rectangles has a good realization: If the 3-SAT formula is satisfiable, then choose the realization that has flow in all horizontal and vertical pipes corresponding to true literals. For the remaining pipes, set all the toggles parallel to the pipe (no flow). For each variable, only one of the two horizontal pipes has flow, so there is a good realization for the two toggles corresponding to the variable. For each clause, at least one of the vertical pipes has flow, so there is a good realization for the elastic rectangle corresponding to the clause. Thus there

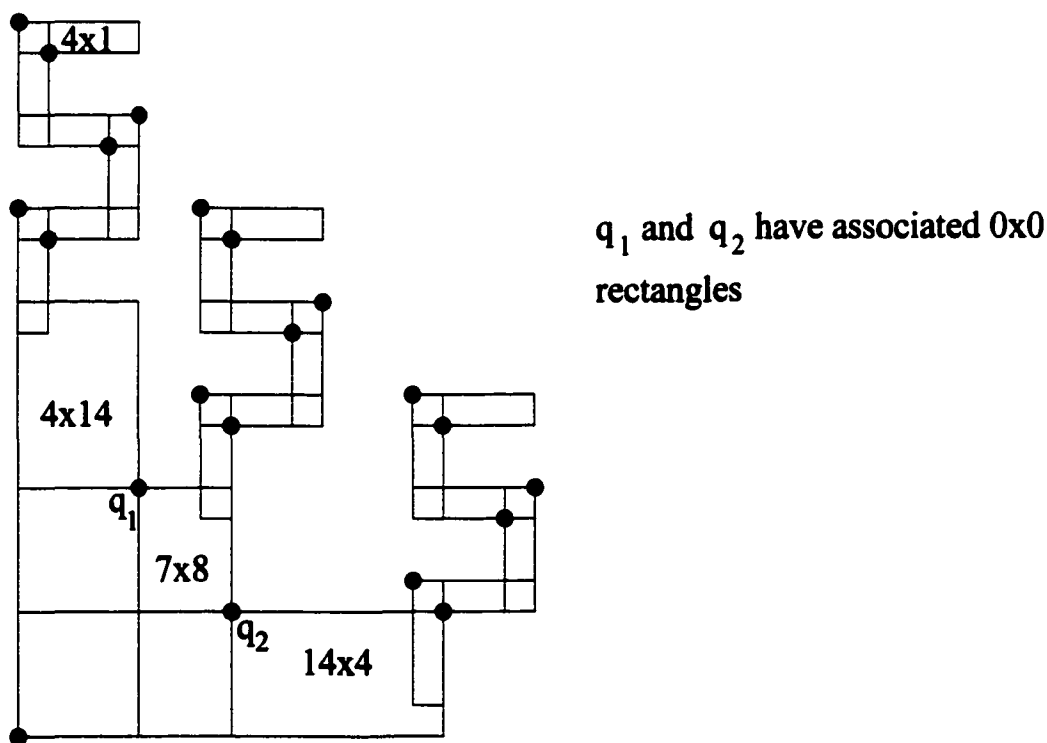


Figure 8.4: Clause gadget.

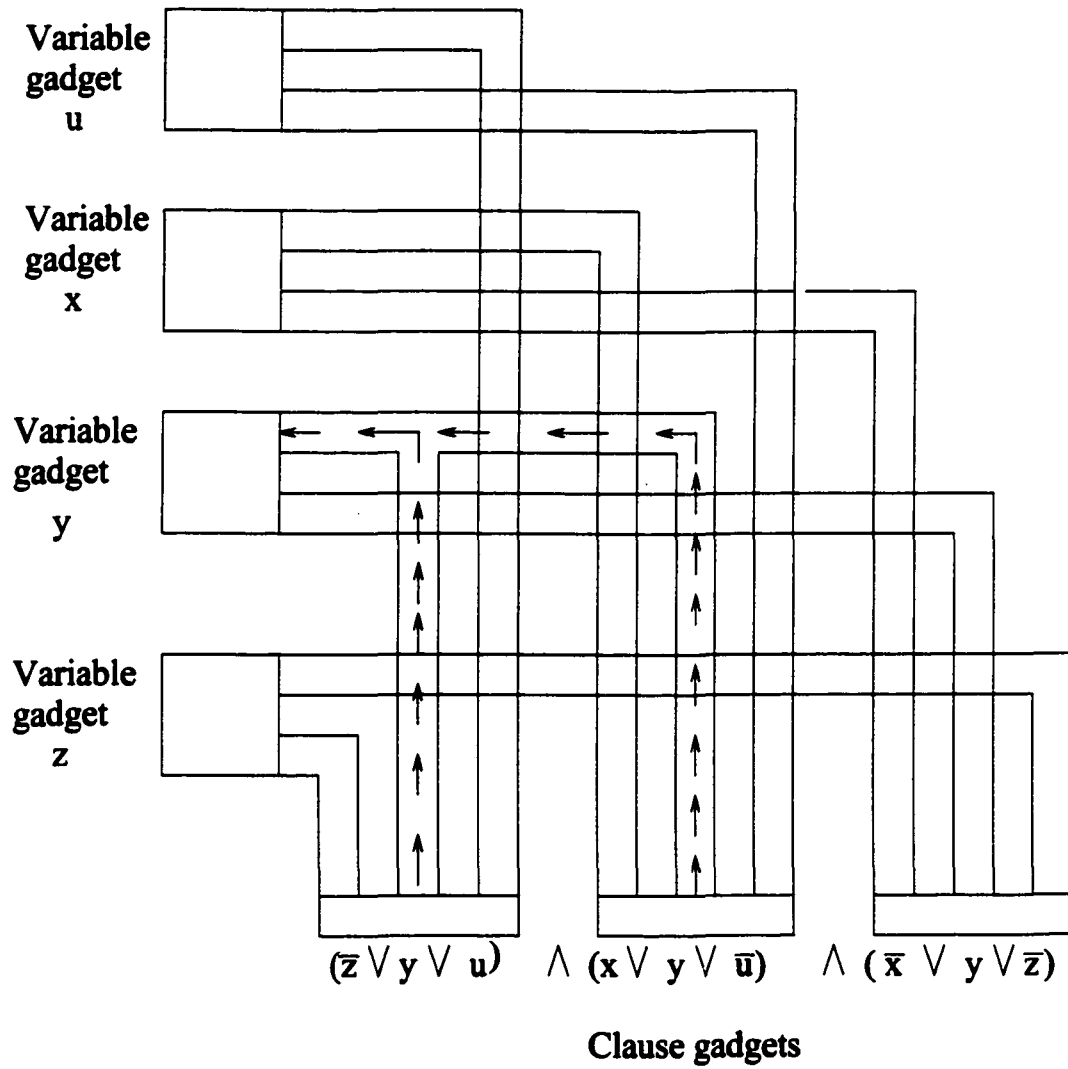


Figure 8.5: Layout of variable, clause, and pipe gadgets.

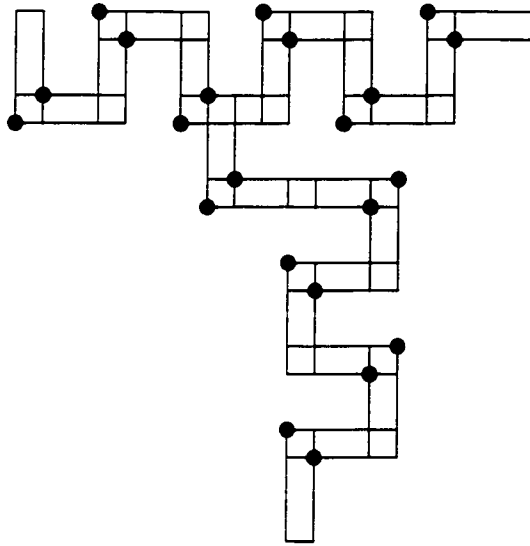


Figure 8.6: Connection between pipes.

is a good realization for the whole set of elastic rectangles. Conversely, if there is a good realization for the elastic rectangles, then set variable x to True if there is flow in horizontal pipe p_x , and set variable x to False otherwise. Since at least one of the vertical pipes leaving each clause gadget has flow in it, and this flow is preserved when the vertical pipe joins the horizontal pipe for that literal, thus this truth-value setting satisfies all the clauses.

It remains to describe the crossing gadget. The horizontal pipe is unchanged, except that one toggle is stretched. In order for the vertical pipe to cross the horizontal one, some toggle of the vertical pipe must intersect some toggle of the horizontal pipe. This causes trouble when we would like to use the intersecting realizations for the two intersecting toggles. The fix is to put in a detour on the vertical pipe, with one branch of the vertical pipe intersecting a vertical realization of a toggle from the horizontal pipe, and the other branch of the vertical pipe intersecting a horizontal realization of a toggle from the horizontal pipe. Since the

horizontal pipe will either use vertical realizations or horizontal realizations, thus one of the two branches of the vertical pipe will be available to transmit flow. We must also provide an alternate, third realization for each of the toggles of the vertical pipe that intersect toggles of the horizontal pipe. These are toggles B and D in Figures 8.7 and 8.8 respectively. These figures show the vertical and horizontal parts of a crossing gadget respectively. A complete crossing gadget is shown in Figure 8.9 where the rectangles in bold represent the vertical part of the crossing gadget and the filled and dashed rectangles the horizontal part. The rectangles with anchors U , W , Y and Z have zero width and height.

Lemma 8.1 *Referring to Figure 8.9, if toggle C is set to horizontal, then toggle A must be set to horizontal too.*

PROOF. Consider the two settings for toggle T . If T is set to horizontal, then B cannot be set to vertical or square, so it must be set to horizontal too, so A must be set to horizontal (see Figure 8.10). On the other hand, if T is set to vertical, then D cannot be set to vertical or square, so it must be set to horizontal, so A must be set to horizontal too (see Figure 8.11). $\square$

This lemma shows that the vertical pipe preserves flow. It remains to observe that there are good realizations corresponding to all four possibilities of flow/no flow in each of the two pipes. (See Figures 8.10, 8.11, 8.12, and 8.13.)

We discuss now the time taken by the reduction of a 3-SAT formula to our instance of the one-corner elastic labeling problem built by gadgets and pipes mentioned above.

It is easy to see that each gadget can be built in constant time. We build a vertical pipe for each literal in the given 3-SAT formula. This construction takes in

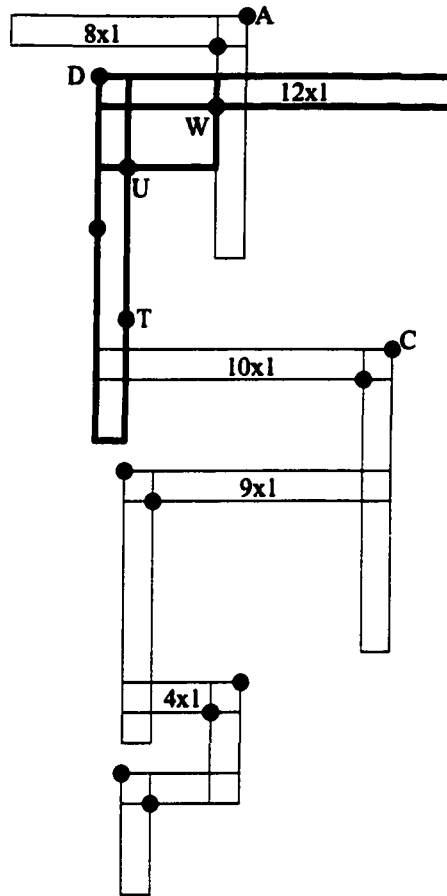


Figure 8.7: Vertical part of the crossing gadget.

the worst case $O(n)$ time, where n is the number of variables in the formula. Then, the time required to build all vertical pipes is $O(nm)$, where m is the number of clauses in the formula. For horizontal pipes we have a similar argument. In consequence the number of crossing gadgets is at most $O(mn)$. The number of connection pipes is linear with respect to the number of clauses. Therefore, the time required for the reduction is polynomial.

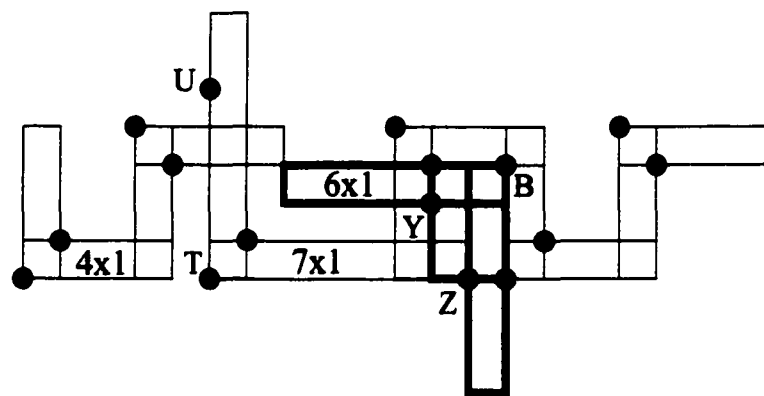


Figure 8.8: Horizontal part of the crossing gadget.

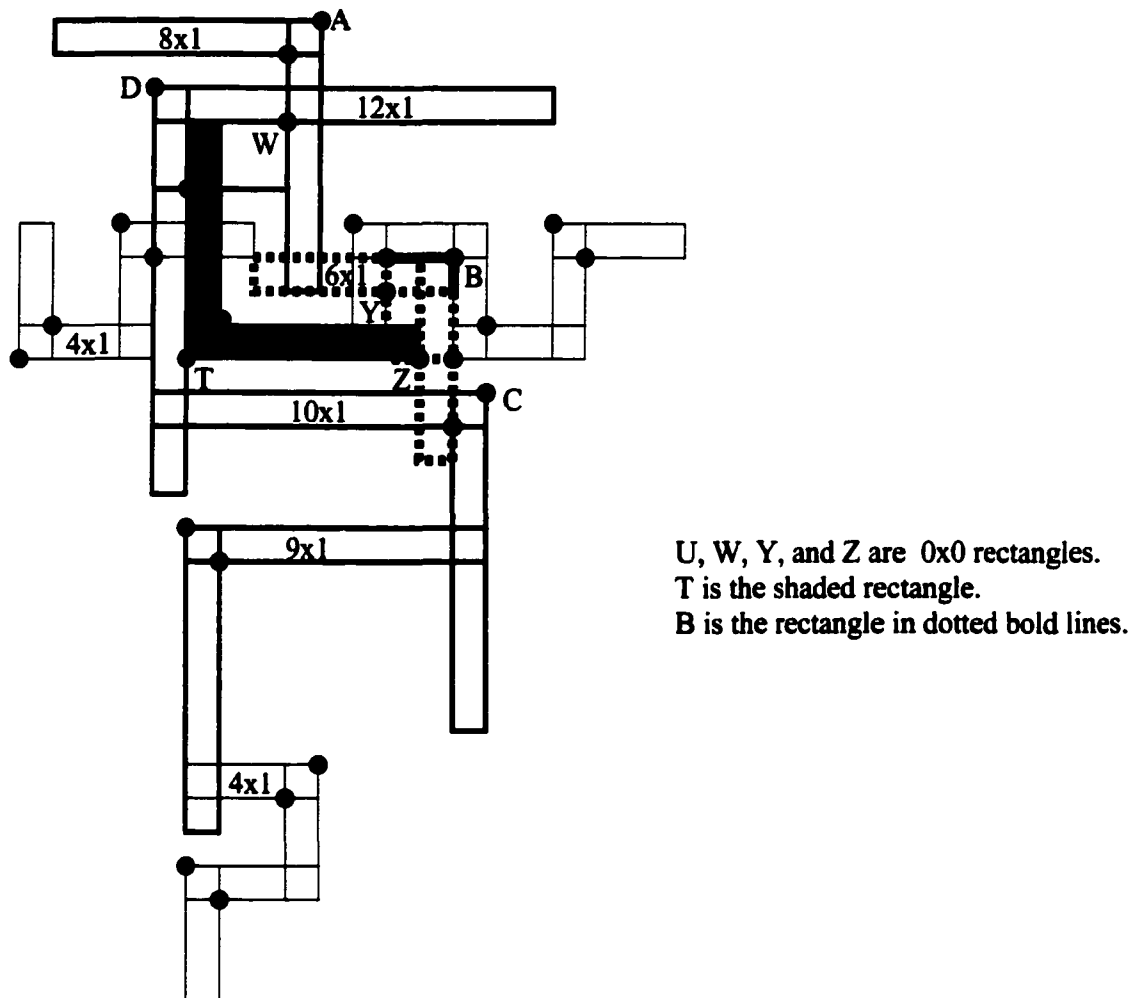


Figure 8.9: Crossing gadget.

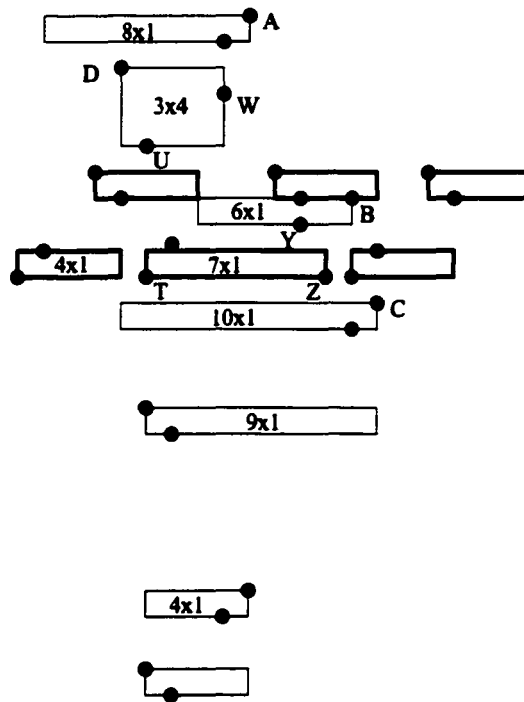


Figure 8.10: Vertical flow only.

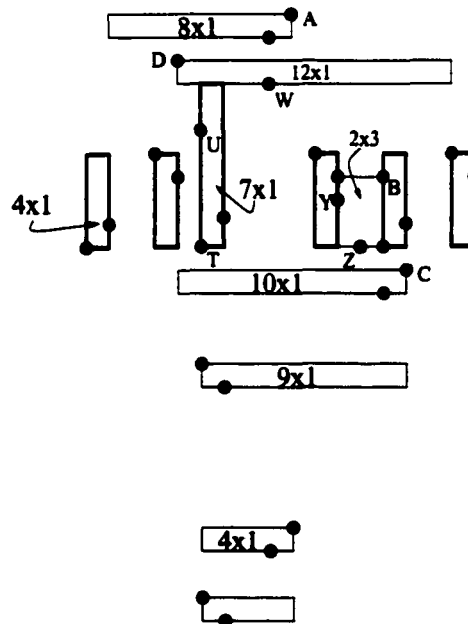


Figure 8.11: Vertical and horizontal flow.

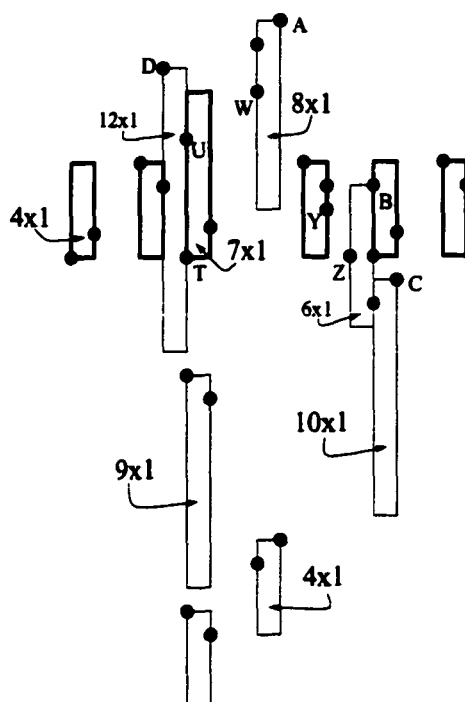


Figure 8.12: Horizontal flow only.

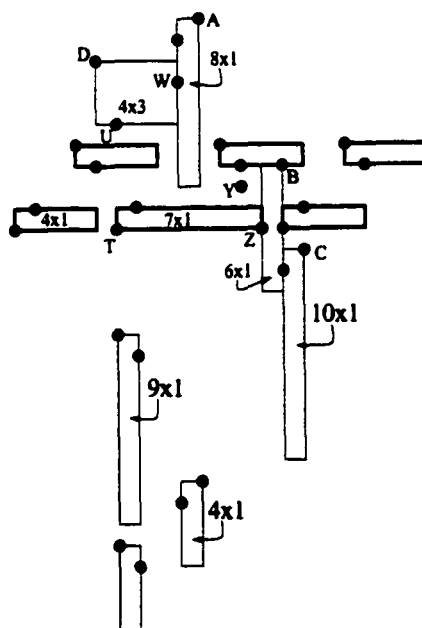


Figure 8.13: Neither vertical nor horizontal flow.

8.2 Sliding Labels Problem

The sliding labels problem is a traditional labeling problem in the sense that the labels are of fixed sizes (not necessarily all of the same size) and we have to position them so that they do not overlap. In this problem, we are given a set of points in the plane, each point associated with an axis-parallel rectangle. The problem is to place the rectangles such that their associated points are on the boundary and the rectangles are pairwise disjoint. For this problem Hirsch [Hir82] gave a heuristic, and van Kreveld et al. [KSW98] provided a 2-approximation algorithm for rectangles of the same height.

If we restrict the associated point to be only on the left or right side of the boundary of its rectangle, we have the *left-right sliding labels problem*. In the following subsection we prove that even the left-right sliding labels problem is NP-hard. Again this proof follows the same strategy given by Formann and Wagner [FWa91].

8.2.1 Left-Right Sliding Labels Problem

We will use notation similar to that of Section 8.1. Rectangles are considered to be topologically open. The point associated with a rectangle is called its *anchor*. A *good realization* for a set of rectangles is a placement of the rectangles at their anchors such that the rectangles are pairwise disjoint. As for the one-corner labeling problem we reduce an instance of the 3-SAT problem to an instance of the left-right labeling problem. We build variable, clause and crossing gadgets, and pipes. The overall structure is exactly the same as before, just details of the gadgets change.

We say a rectangle is *standard* when it has width and height equal to 2.

A *pipe* for this instance is a set of points in the plane associated with either

standard rectangles or with rectangles of zero width and height. We call the anchors with rectangles of zero width and height *dummy points*. As before we introduce two kind of pipes: horizontal and vertical.

In the figures for this subsection, we indicate with an arrow the anchor of a rectangle. The basic building block for a *horizontal pipe* consists of a point A associated with a standard rectangle and four dummy points as illustrated in Figure 8.14a. The dummy points guarantee that the rectangle for A can only be in two possible positions, to the right of A , with A in the middle of the left side, or to the left of A , with A in the middle of the right side. We put several basic building blocks together to form a horizontal pipe. A horizontal pipe is directed from right to left. Note that if the first rectangle in the pipe is to the left of its anchor, then all the rectangles in the pipe are to the left of their anchors. In this case, we say that there is *flow* in the pipe (see Figure 8.15a).

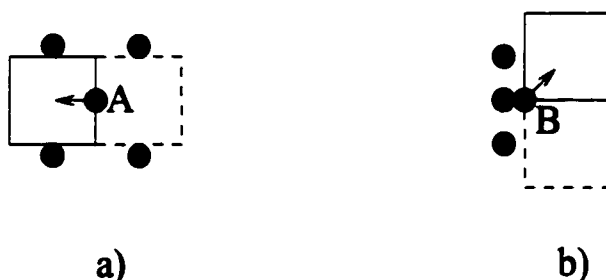


Figure 8.14: Basic blocks.

The basic building block for a *vertical pipe* is illustrated in Figure 8.14b, where point B is associated with a standard rectangle and the other points are dummy points. The dummy points guarantee that the rectangle for B must be to the right of B . We put several of these basic building blocks together to form a vertical pipe. A vertical pipe is directed from the bottom to the top. Note that if the first rectangle of the pipe has its anchor as bottom corner then all the rectangles above

must have their anchors as bottom corners. In this case, we say that there is *flow* in the pipe (see Figure 8.15). Note that, unlike horizontal pipes, there is no unique “no flow” position.

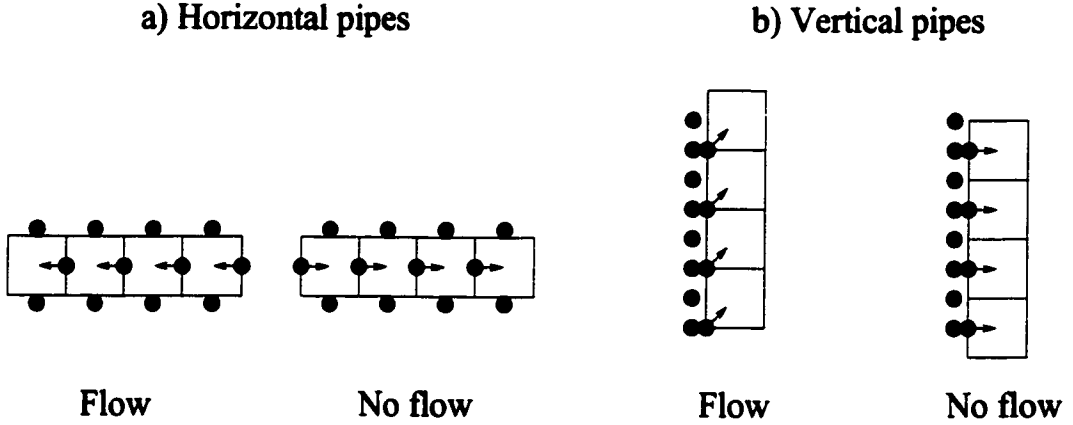


Figure 8.15: Pipes.

For each variable x of a 3-CNF formula we build a *variable gadget* shown in Figure 8.16. This gadget consists of a point associated with a rectangle of width 3 and height 8, dummy points, and two horizontal pipes. We call the horizontal pipe at the top of the variable gadget p_x and the one at the bottom $p_{\bar{x}}$. Observe that there can be flow in at most one of the pipes. Setting variable x True [False] will correspond to having flow in pipe p_x [$p_{\bar{x}}$, respectively].

For each clause we have a *clause gadget*, which consists of a point A associated with a rectangle of height and width equal to 4, point E associated with a standard rectangle, and dummy points B , C , D , and F . Three vertical pipes, V_1 , V_2 , and V_3 are connected to the clause gadget as indicated in Figure 8.17. The dummy points force the anchor A to be at the bottom right of its rectangle or on the left of its rectangle. In the former case there must be flow in pipe V_1 . In the latter case, if the anchor A is at the bottom left of its rectangle then there must be flow in pipe

V_2 and otherwise the rectangle for E is forced to the right and there must be flow in pipe V_3 (see Figure 8.18).

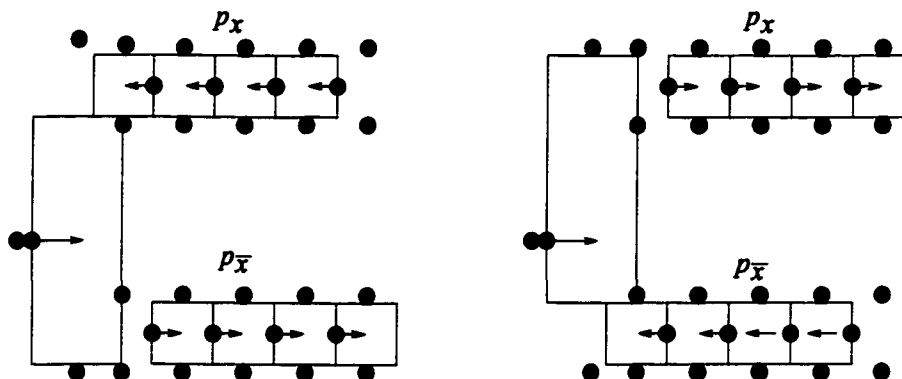


Figure 8.16: Variable gadget.

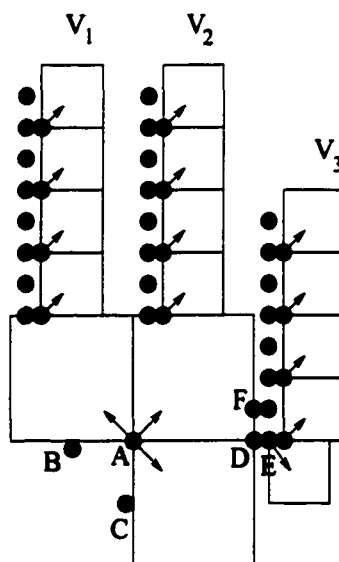


Figure 8.17: Clause gadget.

We connect a vertical pipe with a horizontal pipe as indicated in Figure 8.19. The rectangle R has width 2 and height 3. Observe that if there is flow in the horizontal pipe to the right of R , or flow in the vertical pipe then R must be to the left of its anchor and there must be flow in the horizontal pipe to the left of R . Otherwise, the realization has no flow in either pipe.

We also need crossing gadgets since we have vertical pipes crossing horizontal pipes of other variable gadgets. A *crossing gadget* consists of points A , B , and C as show in Figure 8.20. The rectangle associated with A has width equal to 1 and height equal to 4. Points B and C have rectangles with width equal to 2 and height of 1. We can easily see that if there is flow in the vertical pipe then A is forced to be a bottom anchor (see Figures 8.20a and 8.20b). If A is bottom left corner then this rectangle forces flow in the part of the vertical pipe that continues above the crossing. If A is a bottom right corner then B is forced to be at the left side of its rectangle. In this case the rectangle at B forces flow to continue being transmitted into the vertical pipe above the crossing. When there is flow into the horizontal pipe then the rectangle at A is forced to have A at its right side, thus flow continues in the horizontal pipe to the left of the crossing with the vertical pipe. This case is illustrated in Figures 8.20b and 8.20d. In Figure 8.20c we can see that if there is no flow transmitted by either the vertical or the horizontal pipe then no flow is forced to be transmitted after the crossing.

Finally, we build for an instance of a 3-SAT problem an instance of the left-right sliding labels problem in the same way as the one-corner labeling problem instance as shown in Figure 8.5. We just use the gadgets and pipes for the left-right label problem. The proof for the following claim is similar to the proof of Theorem 8.1.

Claim 8.1 *An instance of 3-SAT problem has a solution iff the instance of the*

left-right sliding labels problem as constructed above has a solution.

With the above claim the following theorem follows.

Theorem 8.2 *The left-right sliding labels problem is NP-hard.*

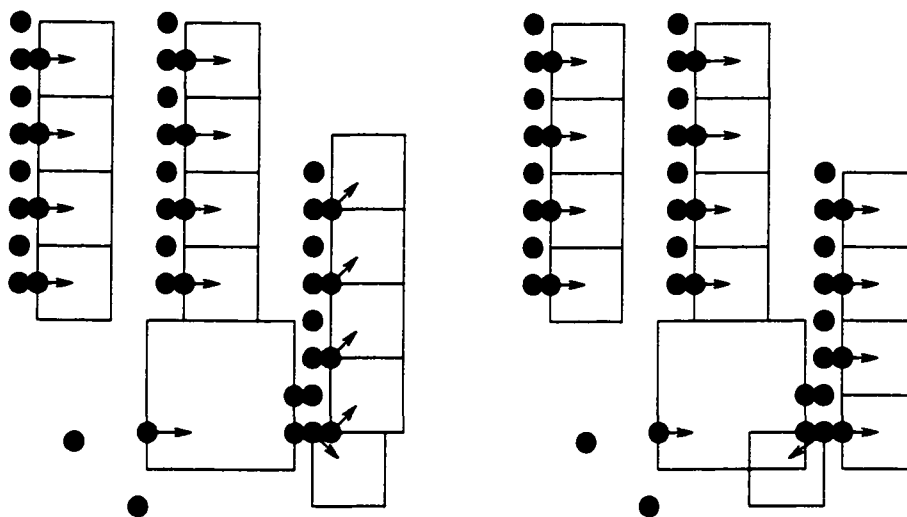


Figure 8.18: Clause gadget forcing flow into pipe V_3 .

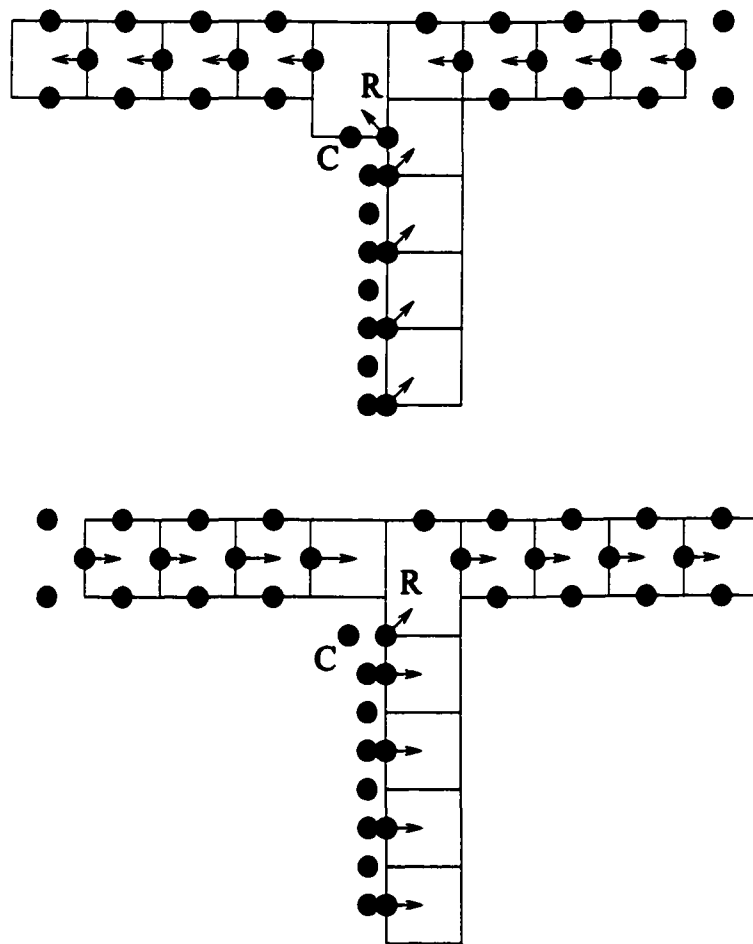


Figure 8.19: Connection of pipes.

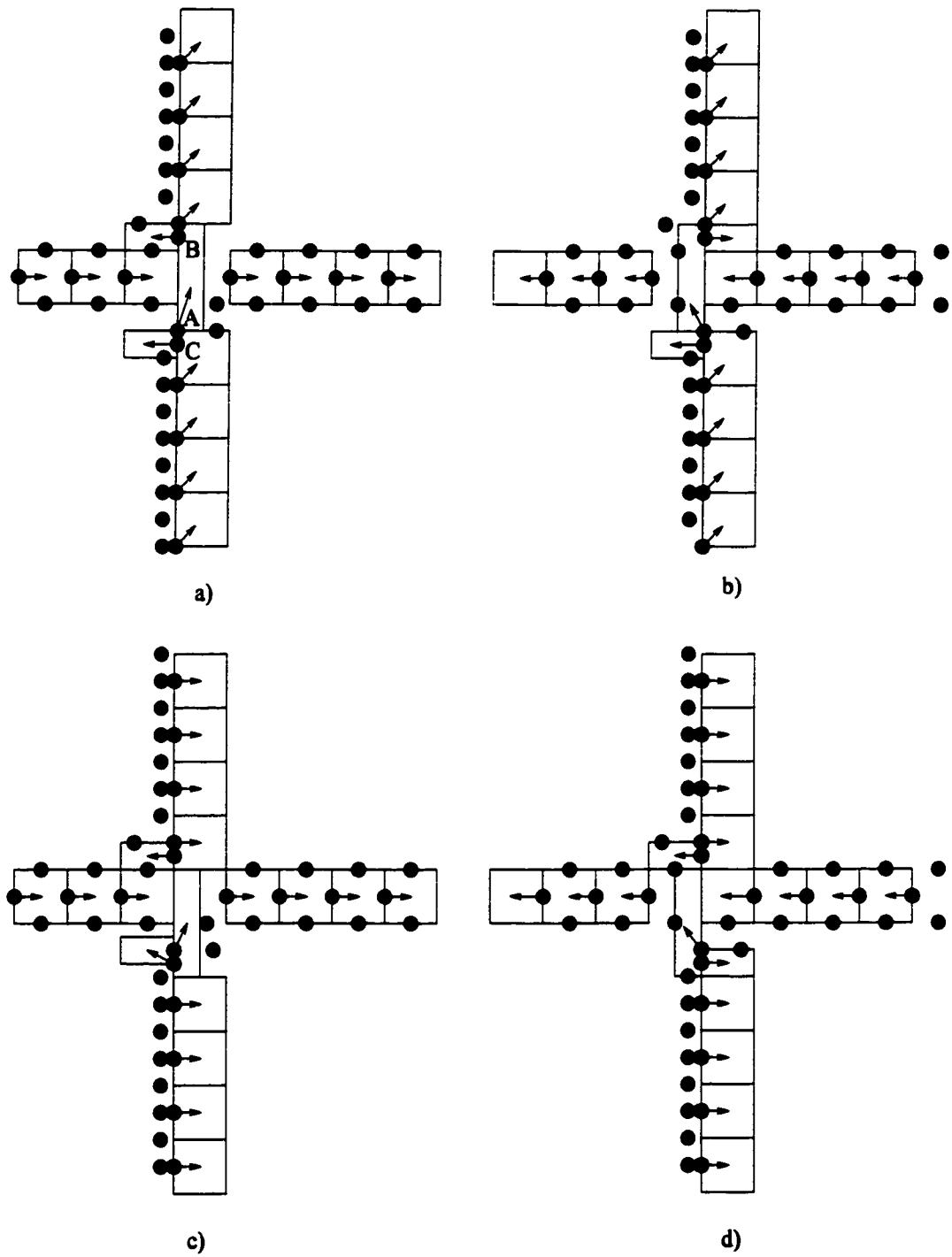


Figure 8.20: Crossing gadget.

8.3 Three-Position Point-Feature Labeling Problem

The point-feature label placement problem was independently shown to be NP-complete by a number of authors [FWa91, KIm88, KRa92, MSh91]. This is the case when a label has four candidate positions, using its associated point as any of its four corners. Formann and Wagner [FWa91] gave an $O(n \log^2 n)$ algorithm when each label has two candidate positions using its associated point in two of its corners.

Then, the question arises, what if we just consider three of the four corners to place a label. We call this problem the *three-position point-feature labeling problem* and it is defined as follows: given a set of points in the plane each associated with an axis-parallel rectangular label, the problem is to place the labels so that they do not overlap, and so that each label has one of the three distinguished corners at its associated point. The three distinguished corners are the same for all points.

In this section, we show that this problem remains NP-complete. The proof follows the strategy given by Formann and Wagner [FWa91]. Since the strategy has been explained with detail in the previous section, we presume that the reader is familiar with it. We refer the reader to Section 8.1 for the terminology used here. We build the corresponding pipes and gadgets using size 2×2 rectangles. The three corners to be considered are indicated in Figure 8.21. Once we have built the pipes and gadgets, we place them together as indicated in Figure 8.5. In all figures for this section we indicate with an arrow the anchor of a rectangle.

A configuration for a vertical and a horizontal pipe is shown in Figure 8.22. A horizontal pipe is directed from right to left, and a vertical pipe from the bottom

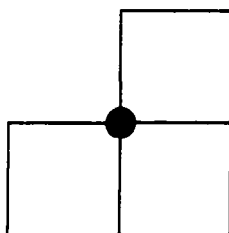


Figure 8.21: Candidate positions for a label.

to the top. Note that in a horizontal pipe, if the first point is a right anchor, then all points in the pipe to its left and in the same height must be right anchors. We say then that there is flow in the pipe. Now, for a vertical pipe if the second bottommost point is a bottom anchor, then all the points in the pipe above it and with the same x -coordinate must be bottom anchors. In this case, there is flow in the vertical pipe.

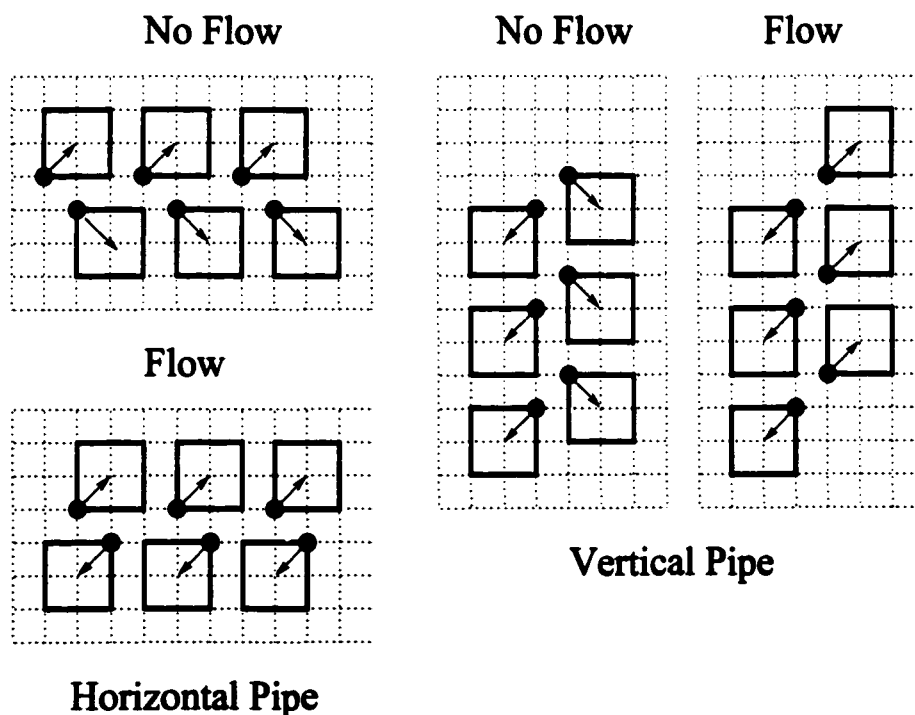


Figure 8.22: Pipes.

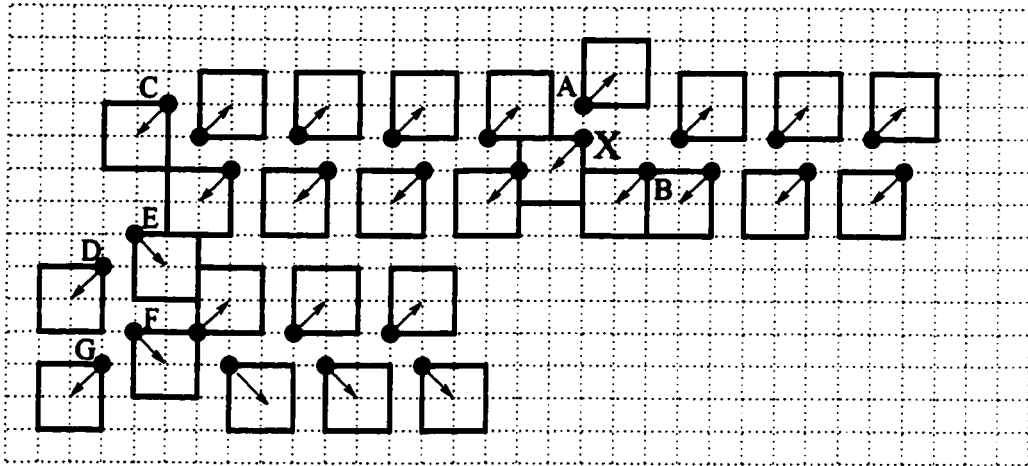


Figure 8.23: Variable gadget set to False.

A variable gadget shown in Figures 8.23, 8.24 and 8.25 consists of two horizontal pipes one above the other. The horizontal pipe at the top is divided by points A , B , and X . Point X represents variable x , a variable of a 3-CNF formula, and the position of its label indicates the truth value of x . These two horizontal pipes are joined at the left with points C , D , E , F , and G as indicated in the figure. When there is flow in the top pipe, X is forced to be either top left anchor (see Figure 8.23) or bottom right anchor (see Figure 8.24) and no flow is transmitted to the bottom pipe. In this case, we say that variable x is set to False. If there is flow in the bottom pipe, then no flow is transmitted in the top pipe and X is forced to be a bottom left anchor. We say that x is set to True.

We connect variable gadgets to the corresponding clause gadgets by using vertical pipes. We connect variable gadgets with vertical pipes as indicated in Figures 8.26 and 8.27.

A clause gadget is shown in Figures 8.28, 8.29, and 8.30. We can see that to have a good realization, flow must be transmitted by at least one of the vertical

pipes connected to the clause gadget.

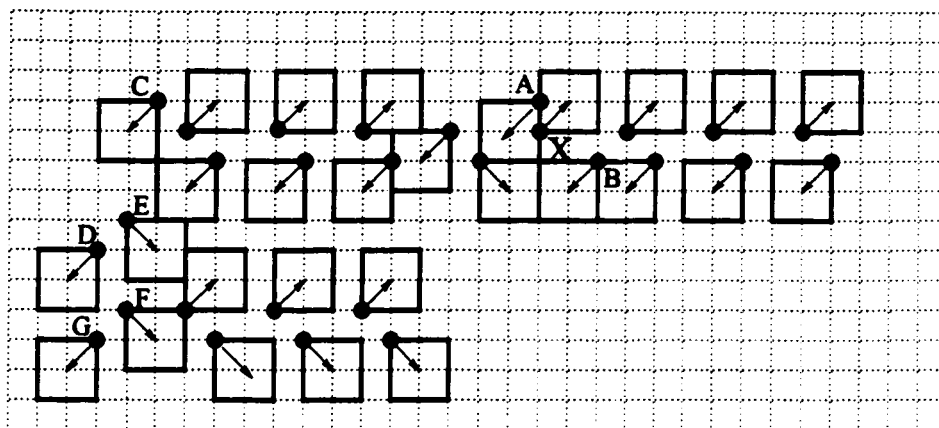


Figure 8.24: Variable gadget set to False.

Finally, in Figure 8.31 we have the configuration for the crossing gadgets with the different kinds of flows that can be transmitted.

Using the pipes and gadgets in the same way as we did in the proof of Theorem 8.1, we get the following theorem.

Theorem 8.3 *The three-position point-feature labeling problem is NP-complete.*

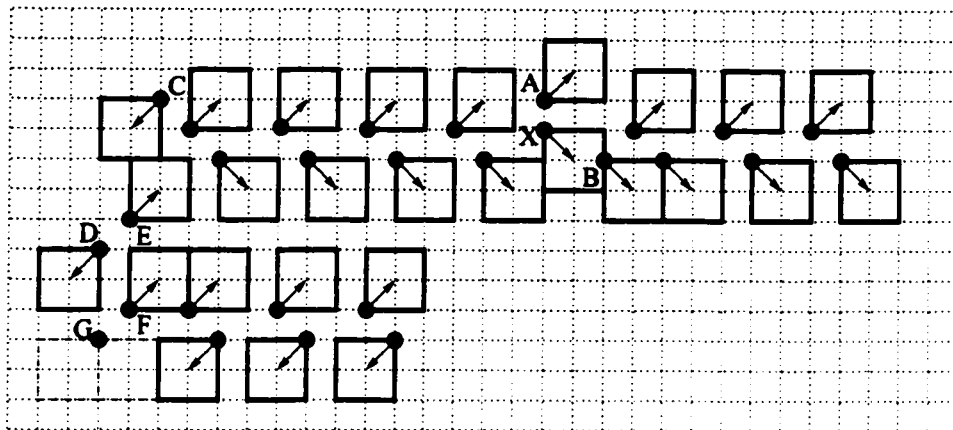


Figure 8.25: Variable gadget set to True.

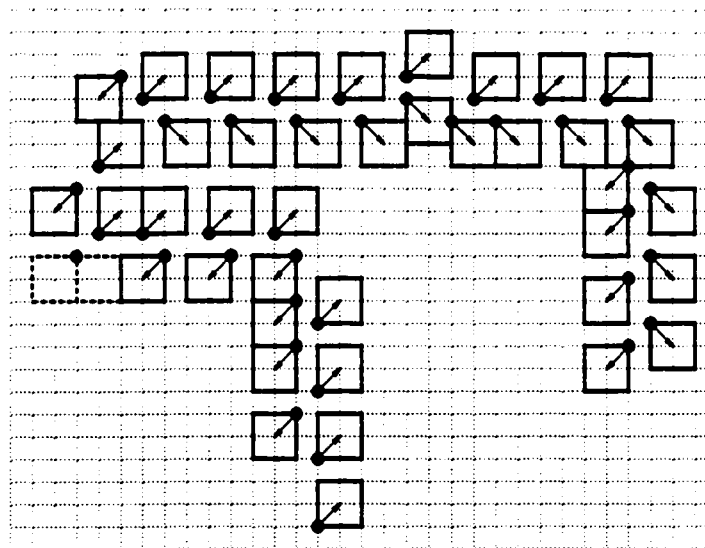


Figure 8.26: Vertical pipes connected to a variable gadget set to True.

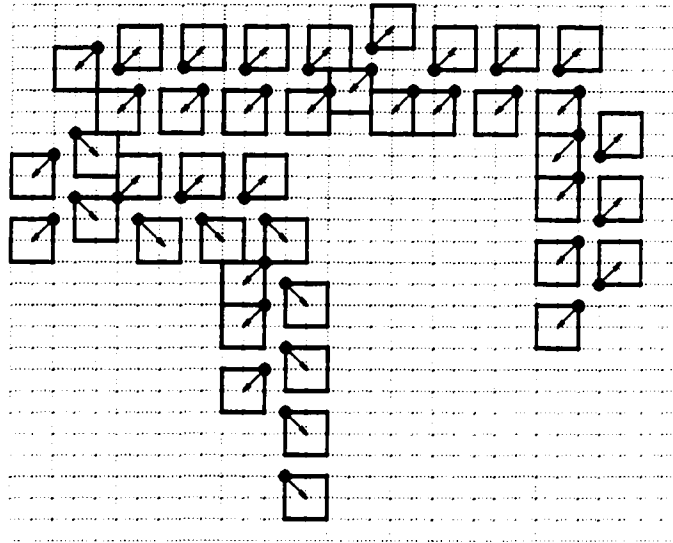


Figure 8.27: Vertical pipes connected to a variable gadget set to False.

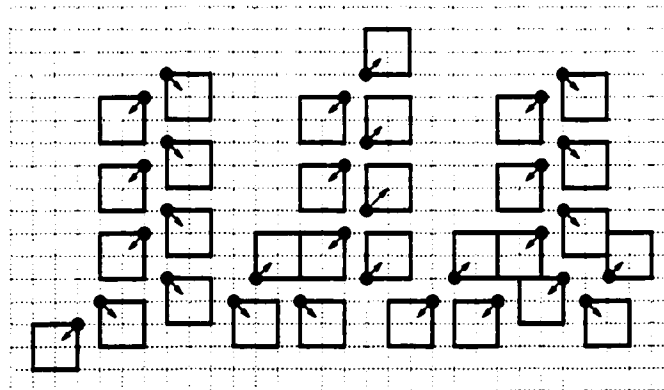


Figure 8.28: Clause gadget.

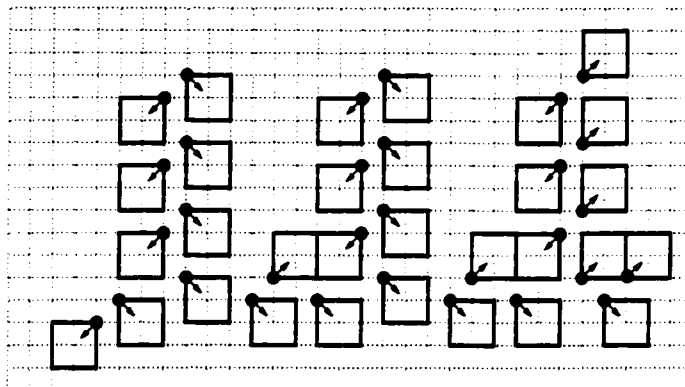


Figure 8.29: Clause gadget.

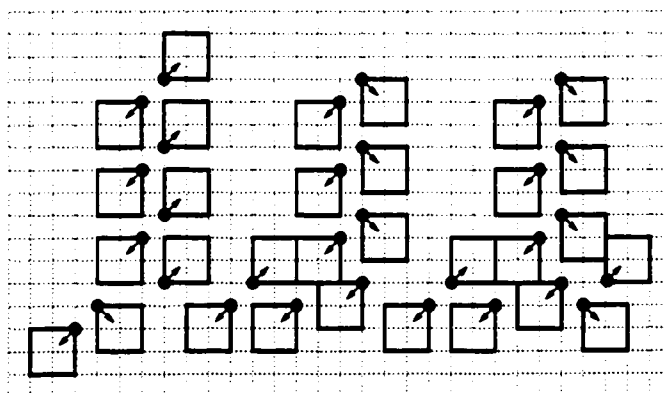


Figure 8.30: Clause gadget.

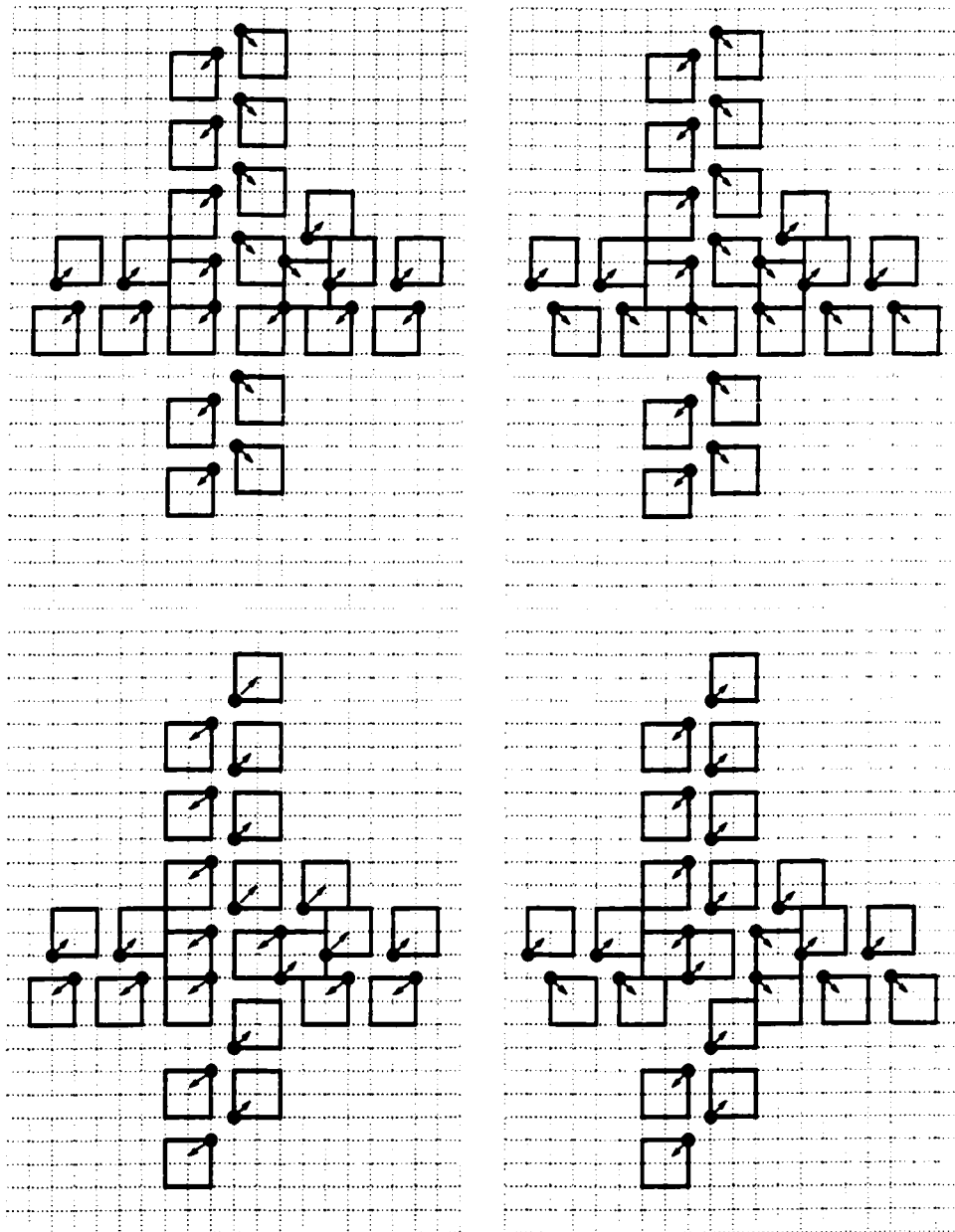


Figure 8.31: Crossing Gadgets.

Chapter 9

Conclusions

In this thesis, we proposed and studied an interesting alternative to the point-feature label placement problem, the *elastic labeling problem*. We formulated this alternative by introducing *elastic labels*. In contrast with the traditional labels, elastic labels offer more options for the label size of each object.

We also studied another variation of the point-feature label placement problem: the *sliding labels problem*. In this problem each label has fixed dimension but any boundary point (not just a corner) of the rectangular label can be placed at the associated point in the plane.

In Chapter 8, we show that even simple variations of these problems remain NP-hard. The variations presented in that chapter were: the *one-corner elastic labeling problem*, where the corner of an elastic label is fixed; and the *left-right sliding labels problem*, where the side of a non-elastic label to slide in its associated point is fixed. As well, we showed that the *three-position point-feature labeling problem* is NP-complete. This problem is a variation of the point-feature label placement problem where labels have three candidate positions.

In Chapter 6, we gave polynomial time algorithms for the *same-corner elastic labeling problem*, a case of the elastic labeling problem where all labels have the same corner fixed, and for the *same-side sliding labels problem*, a case of the sliding labels problem where all labels slide in its associated point on the same side.

In Chapter 7, we presented two approximation algorithms for the point-feature label placement problem. One algorithm is for the case where the labels are squares not necessarily of the same size. The other algorithm is for labels that are rectangles with weight associated to them, and the algorithm favors rectangles of big sizes.

In Chapters 3, 4, and 5 we concentrated in several variations of the elastic labeling problem.

In Chapter 3, we studied the *two-axis labeling problem*, where the labels lie in the first quadrant on the positive x and y axes. We gave a polynomial time dynamic programming algorithm for this problem.

In Chapter 4, we studied the *two-parallel lines labeling problem*, where the labels lie on two parallel lines. We also gave a polynomial time algorithm for this problem.

The results presented in Chapter 3 and Chapter 4 were used to obtain the main result of the thesis. In Chapter 5, we presented this result: a polynomial time algorithm for the elastic labeling problem when the given points lie on a boundary of a rectangular map—the *rectangle perimeter labeling problem*. In Chapter 2, we discussed relevant applications of this problem. In particular, we discussed its potential applications in software for Geographical Information Systems (GIS) and for Global Positioning Systems (GPS).

In recent years, with the introduction of the web and GIS dynamic maps are becoming more and more accessible to the public. In those, we have maps that are clipped to fit their rectangular screens. Thus, we have provided an algorithm to

dynamically place labels on the boundary of those maps. We suggested the use of these labels to help navigate a map. In the labels we can place information about what is beyond the boundary indicating what place we would reach if we follow a direction beyond the boundary.

It is worth mentioning that, although we emphasized the labeling of maps, the techniques can be used for any diagram such as charts and histograms.

Bibliography

- [AKS98] P.K. Agarwal, M. van Kreveld, and S. Suri. Label placement by maximum independent set in rectangles. In *Proc. 9th Canadian Conference on Computational Geometry*. (1997), 233–238.
- [AFr84] J. Ahn and H. Freeman. A program for automatic name placement. *Cartographica*. **21** (2 & 3) (1984), 101–109.
- [AIK89] H. Aonuma, H. Imai, and Y. Kambayashi. A visual system of placing characters appropriately in multimedia map databases. In *Proc. of the IFIP TC 2/WG 2.6 Working Conference on Visual Database Systems*. North-Holland (1989), 525–546.
- [AKa91] M. Arikawa and Y. Kambayashi. Dynamic name placement functions for interactive map systems. *The Australian Computer Journal*. **23** 4 (1991), 133–147.
- [AKK94] M. Arikawa, H. Kawakita, and Y. Kambayashi. Dynamic maps as composite views of varied geographic database servers. In *Proc. of the First International Conference on Applications of Databases*. Volume 819 of Lecture Notes in Computer Science. Springer-Verlag (1994), 142–157.

- [BSu94] M. Bellare and M. Sudan. Improved non-approximability results. In *Proc. 26th ACM Symposium on the Theory of Computing*. (1994), 184–193.
- [BKO97] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry-Algorithms and Applications*. Springer-Verlag, Berlin, 1997.
- [BEp97] M. Bern and D. Eppstein. Approximation algorithms for geometric problems. In Dorit S. Hochbaum, editor, *Approximation algorithms for NP-hard Problems*. PWS Publishing Company, Boston, MA. (1997), 296–345.
- [BCr94] D. Beus and D. Crockeit. Automated production of 1:24,000 scale quadrangle maps. In *Proc. 1994 ASPRS/ACSM Ann. Convention and Exposition. 1* (1994), 94–99.
- [Bon57] W. Bonacker. *Die Namenstellung in geographischen Karten*. Haller (studien zur Kartographie 1 a), Berlin, (1957).
- [Cer85] V. Cerny. A thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory Applications*. 45 (1985), 41–51.
- [Cha96] B.Chazelle. Application challenges to computational geometry: CG impact task force report. *Technical Report TR-521-96*. Princeton University, April 1996.
- [CMS93] J. Christensen, J. Marks, and S. Shieber. Algorithms for cartographic label placement. In *Proc. of the American Congress on Surveying and Mapping 1*. (1993), 75–89.

- [CMS94] J. Christensen, J. Marks, and S. Shieber. Placing text labels on maps and diagrams. In Paul Heckbert, editor, *Graphics Gems IV*. Academic Press, Boston, MA (1994), 497–504.
- [CMS95] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point feature label placement. *ACM Transactions on Graphics*. **14** 3 (1995), 203–232.
- [CMS97] J. Christensen, S. Friedman, J. Marks, and S. Shieber. Empirical testing of algorithms for variable-sized label placement. *Proc. 13th ACM Symposium on Computational Geometry*. (1997), 415–417.
- [CJo90] A.C. Cook and C.B. Jones. A prolog rule-based system for cartographic name placement. *Computer Graphics Forum*. **9** (1990), 109–126.
- [Den94] E. A. Dengler. *Perceptual properties of graph layouts*. Master Thesis in Computer Science, University of Waterloo, 1994.
- [DET94] G. Di Battista, P. Eades, R. Tamassia, and I. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry: Theory and Applications*. **4** (1994), 235–282.
- [Den96] B. D. Dent *Cartography : thematic map design*. Dubuque, IA : Wm. C. Brown, 1996.
- [DMM97] S. Doddi, M. V. Marathe, A. Mirzaian, B. M. Moret, and B. Zhu. Map Labeling and its generalizations. In *Proc. 8th ACM-SIAM Symposium on Discrete Algorithms*. (1997), 148–157.

- [DFr92] J.S. Doerschler and H. Freeman. A rule-based system for dense-map name placement. *Communications of the Association of Computing Machinery*. **35** 1 (1992), 68–79.
- [EGo90] L.R. Ebinger and A.M. Goulete. Noninteractive automated names placement for the 1990 decennial census. *Cartography and Geographic Information Systems*. **17** 1 (1990), 69–78.
- [Ede83] H. Edelsbruner. A new approach to rectangle intersections, Part I. *Internat. J. Comput. Math.* **13** (1983), 209–219.
- [Ede83-2] H. Edelsbruner. A new approach to rectangle intersections, Part II. *International Journal of Computer Mathematics*. **13** (1983), 221–229.
- [ECM97] S. Edmondson, J. Christensen, J. Marks, and S.M. Shieber. A general cartographic labeling algorithm. *Cartographica*. **33** 4 (1997), 13–23.
- [EIS76] S. Even, A. Itai and A. Shamir. On the complexity of time table and multicommodity flow problem. *SIAM Journal on Computing*. **5** (1976), 691–703.
- [FPl99] J-D. Fekete and C. Plaisant. Dynamic Neighborhood Labeling for Data Visualization. In *Proc. of ACM CHI'99: Human Factors in Computing Systems*. (1999), 512–519.
- [FWa91] M. Formann and F. Wagner. A packing problem with applications in lettering of maps. In *Proc. 7th ACM Symposium on Computational Geometry*. (1991), 281–288.

- [FPT81] R.J. Fowler, M.S. Paterson, and S.L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters*. **12** 3 (1981), 133–137.
- [Fre91] H. Freeman. Computer name placement. In D.J. Maguire, M.F. Goodchild, and D.W. Rhind, editors, *Geographical Information Systems: Principles and Applications*. Longman London, 1991.
- [FrA87] H. Freeman and J. Ahn. On the problem of placing names in a geographical map. *International Journal of Pattern Recognition and Artificial Intelligence*. **1** (1) (1987), 121–140.
- [GJS96] P. Gupta, R. Janardan, and M. Smid. Efficient algorithm for counting and reporting pairwise intersections between convex polygons. In *Proc. 8th Canadian Conference on Computational Geometry*. (1996), 8–13.
- [Hir82] S.A. Hirsch. An algorithm for automatic name placement around point data. *The American Cartographer*. **9** 1 (1982), 5–17.
- [HMa85] D.S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*. **32** (1985), 130–136.
- [Hoc97] D. S. Hochbaum. *Approximation Algorithms for NP-hard problems*. PWS Publishing Company, 1997.
- [HTa92] J. Y. Hsiao and C. Y. Tang. An Efficient Algorithm for Finding a Maximum Weight 2-Independent Set on Interval Graphs. *Information Processing Letters*. **43** 5 (1992), 229–235.

- [HMR95] H.B. Hunt III, M.V. Marathe, V. Radhakrishnan, S.S. Ravi, D.J. Rosenkrantz, and R.E. Stearns. NC-Approximation Schemes for NP- and PSPACE-Hard Problems for Geometric Graphs. *Journal of Algorithms*. **26** (1998), 238–274.
- [HMS95] R. Hwa, J. Marks and S. Shieber. Automatic Structuring of Embedded Hypermedia Documents. *Technical Report TR-95-6*. A Mitsubishi Electric Research Laboratory, 1995.
- [IGN34] Institut Geographique National. *Les Ecritures sur les cartes topographiques*. Paris (1934).
- [Ima82] H. Imai. Finding connected components of an intersection graph of squares in the Euclidean plane. *Information Processing Letters*. **15** 3 (1982), 125–128.
- [IAs83] H. Imai and T. Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *Journal of Algorithms*. **4** (1983), 310–323.
- [Imh75] E. Imhof. Positioning names on maps. *The American Cartographer*. **2** (1975), 128–144.
- [ILu97] C. Iturriaga and A. Lubiw. NP-hardness of some map labeling problems. *Technical Report CS-97-18*. University of Waterloo, 1997.
- [ILu97-2] C. Iturriaga and A. Lubiw. Elastic labels: the two-axis case. In G. Di Battista editor, *Graph Drawing (Proc. GD'97)*. vol. 1353 of *Lecture Notes in Computer Science*. Springer-Verlag. (1998), 181–192.

- [ILu98] C. Iturriaga and A. Lubiw. Elastic Labels on the Perimeter of a Rectangle. In S.H. Whitesides editor, *Graph Drawing (Proc. GD'98)*. vol. 1547 of *Lecture Notes in Computer Science*. Springer-Verlag. (1998), 452–453.
- [ILu99] C. Iturriaga and A. Lubiw. Elastic Labels Around the Perimeter of a Map. To appear in Proc. of Workshop Algorithms and Data Structures, August 1999.
- [ISr89] E.H. Isaaks and R. M. Srivastava. *An introduction to Applied Geostatistics*. Oxford University Press, New York, 1989.
- [Jol96] P. Jolly. Naming places, placing names. *Mapping Awareness*. 10 (1996), 28–30.
- [Jon89] C. B. Jones. Cartographic name placement with prolog. *IEEE Computer Graphics & Applications*. (1989), 36–47.
- [KTo96] K. Kakoulis and I.G. Tollis. On the edge label placement problem. In S. North editor, *Graph Drawing (Proc. GD'96)*. vol. 1190 of *Lecture Notes in Computer Science*. Springer-Verlag. (1997), 241–256.
- [KTo97] K. Kakoulis and I.G. Tollis. An algorithm for labeling edges of hierarchical drawings. In G. Di Battista editor, *Graph Drawing (Proc. GD'97)*. vol. 1353 of *Lecture Notes in Computer Science*. Springer-Verlag. (1998), 169–180.
- [KTo98] K. Kakoulis and I. Tollis. A unified approach to labeling graphical features. In *Proc. 14th ACM Symposium on Computational Geometry*. (1998), 347–356.

- [KTo98-2] K. Kakoulis and I. Tollis. On the multiple label placement problem. *Electronic proceedings 10th Canadian Conference on Computational Geometry.*, 1998, <http://cgm.cs.mcgill.ca/cccg98>.
- [KIm88] T. Kato and H. Imai. The NP-completeness of the character placement problem of 2 or 3 degrees of freedom. *Record of Joint Conference of Electrical and Electronic engineers in Kyushu.* (1988), 11–18. In Japanese.
- [KGV83] S. Kirkpatrick, C.D. Gelatt Jr, and M.P. Vecchi. Optimization by simulated annealing. *Science.* 220 (1983), 671–680.
- [KPl81] D. Knuth and M. Plass. Breaking Paragraphs into Lines. *Software-Practice and Experience.* 11 (1981), 1119–1184.
- [KR92] D. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM Journal on Discrete Mathematics.* 5 3 (1992), 422–427.
- [KSW98] M. van Kreveld, T. Strijk, and A. Wolff. Point set labeling with sliding labels. In *Proc. 14th ACM Symposium on Computational Geometry.* (1998), 337–346.
- [KMP93] L. Kučera, K. Mehlhorn, B. Preis, and E. Schwarzenacker. Exact algorithms for a geometric packing problem. In *Proc. 10th Symposium Theoret. Aspects Comput. Sci.*, volume 665 of Lecture Notes Comput. Sci., pages 317–322. Springer-Verlag, 1993.
- [MHR92] M.V. Marathe, H.B. Hunt III, and S.S. Ravi. Geometry based approximations for intersection graphs. In *Proc. 4th Canadian Conference on Computational Geometry.* (1992), 244–249.

- [MSh91] J. Marks and S. Shieber. The computational complexity of cartographic label placement. *Technical Report CRCT-05-91*. Harvard University, 1991.
- [PZC98] C.K. Poon, B. Zhu, and F. Chin. A polynomial time solution for labeling a rectilinear map. *Information Processing Letters*. **65** 4 (1998), 201–207.
- [PSH85] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.
- [RMG95] A.H. Robinson, J.L. Morrison, P.C. Muehrcke, A.J. Kimerling and S.C. Guptill. *Elements of Cartography*. John Wiley & Sons, Inc. 6th edition (1995), 404–424.
- [Tyn92] J. Tyner. *Introduction to Thematic Cartography*. Prentice Hall, 1992, 233–248.
- [Wag94] F. Wagner. Approximate the map labeling is in $\Omega(n \log n)$. *Information Processing Letters*. **52** (1994), 161–165.
- [WWo95] F. Wagner and A. Wolff. Map labeling heuristics: provably and practically useful. In *Proc. 11th ACM Symposium on Computational Geometry*. (1995), 109–118.
- [WWo95-2] F. Wagner and A. Wolff. An efficient and effective approximation algorithm for the map labeling problem. In *Proc. 3rd. Ann. European Symposium of Algorithms*. (1995), 420–433.
- [WWo95-3] F. Wagner and A. Wolff. Fast and reliable map labeling. In *Proc. 9th International Symposium on Computer Science for Environment Protection (CSEP 95)*. (1995), 667–675.

- [WWo97] F. Wagner and A. Wolff. A practical map labeling algorithm. *Computational Geometry: Theory and Applications*. (1997), 387–404.
- [WWo98] F. Wagner and A. Wolff. A combinatorial framework for map labeling. In S.H. Whitesides editor, *Graph Drawing (Proc. GD'98)*. vol. 1547 of *Lecture Notes in Computer Science*. Springer-Verlag. (1998), 316–331.
- [Wol95] A. Wolff. Map labeling. *Master's thesis*. Fachbereich Mathematik und Informatik, Freie Universitdt Berlin, 1995.
- [WSt96] A. Wolff and T. Strijk.. A map labeling bibliography, 1996.
<http://www.inf.fu-berlin.de/map-labeling/papers.html>.
- [Woo87] D. Woodward. *Art & Cartography*. The University of Chicago Press (1987), 174–212.
- [Yoe72] P. Yoeli. The logic of automated map lettering. *The Cartographic Journal*. **9** (1972), 99–108.
- [Zor86] S. Zoraster. Integer Programming applied to the map label placement problem. *Cartographica*. **23** 3 (1986), 16–27.
- [Zor90] S. Zoraster. The solution of large 0-1 integer programming problems encountered in automated cartography. *Operations Research*. September-October **38** 5 (1990), 752–759.
- [Zor98] S. Zoraster. Practical results using simulated annealing for point feature label placement. *Cartography and GIS*. **24** 4 (1997), 228–238.