15

Labeling Algorithms

Konstantinos G. Kakoulis
T.E.I. of West Macedonia, Greece

Ioannis G. Tollis
University of Crete, Greece

15.1 Introduction 489
15.2 The Labeling Problem 490
Searching for a Good Label Assignment • A Definition of the Labeling Problem
15.3 Solving the Labeling Problem 492
The GFLP Problem • The ELP Problem • The NLP Problem • The MLP Problem • Placing Labels by Modifying the Drawing
References 513

15.1 Introduction

An important aspect of information visualization is the automatic placement of text or symbol labels corresponding to graphical features of drawings and maps. Labels are textual descriptions that convey information or clarify the meaning of complex structures presented in a graphical form. The automatic label placement problem is identified as an important research area by the ACM Computational Geometry Task Force [C+99]. It has applications in many areas including cartography [RMM+95], geographic information systems [Fre91], and graph drawing [DETT99].

Because the labeling process is a monotonous and very demanding task, its automation is very desirable. It is very difficult to quantify all the characteristics of a good label placement since they reflect human visual perception, intuition, and experience, which have been perfected through the centuries by cartographers who have elevated the placement of labels into an art. Hence, it is unlikely that computer-based systems will be able to deliver fully automated placement of labels in maps of a sufficient quality to be comparable to those produced manually by experienced cartographers. Nevertheless, there are many areas where the requirements for high aesthetic quality are not as strict and automatic labeling techniques may be applied. For example, these techniques may be used for real time name placement in the context of on-line geographic information systems or internet-based map search, and special-purpose maps such as those used to display census [EG90], oil exploration [Zor90] or soil survey data [FMC96]. Additionally, semi-automated interactive name placement systems may be the most practical approach at the present time. Labeling systems may produce an initial label placement that could be improved manually by cartographers to produce desirable results. Furthermore, the whole concept of map labeling may change depending on computer capabilities [RMM+95]. Maps may be viewed in an electronic format allowing the user to interact and display information on demand as opposed to viewing all the map information at once.

In the following sections, we study the labeling problem not only in its traditional form (i.e., cartography), but also in the context of information visualization, specifically as it relates to graph drawing. In Section 15.2 we present a model for the labeling problem: we discuss the qualities of good label assignment and give a formal definition of the problem. In Section 15.3, we present a variety of algorithms for the labeling problem. Finally, we discuss how one can modify a drawing to accommodate the placement of labels in Section 15.3.5.

15.2 The Labeling Problem

15.2.1 Searching for a Good Label Assignment

Let Γ be a drawing and F be the set of graphical features of Γ to be labeled. A solution to the labeling problem for drawing Γ assigns text or symbol labels to each graphical feature f of F such that the relevant information is communicated in the best possible way. This can be achieved by positioning the labels in the most appropriate places. For each graphical feature there is a large number of potential label positions, and the most preferable among them must be assigned.

Good label placement aids in conveying information and enhances the aesthetics of the input drawing. It is difficult to quantify all the characteristics of a good label placement, because they reflect human visual perception and intuition. It is trivial to place a label when its associated object is isolated. The real difficulty arises when the freedom to place a label is restricted by the presence (in close proximity) of other objects of the drawing. In this common scenario, we must consider not only the position of a label with respect to its associated object, but also how it relates to other labels and objects in the surrounding area.

In a successful label assignment, labels must be positioned such that they are legible and follow basic aesthetic quality criteria. According to cartographers like Imhof [Imh75] and Yoeli [Yoe72], who have extensively studied this subject, labels must be placed in the best position available following some basic rules: Labels must be easily read, quickly located, a label and the object to which it belongs should be easily recognized, labels must be placed very close to the objects they belong to, labels must not obscure other labels or objects, a label must be placed in the most preferred position, among all legible positions. We summarize the labeling quality evaluation in the following three basic rules:

- No overlaps of a label with other labels or other graphical features of the drawing are allowed.
- Each label can be easily identified with exactly one graphical feature of the drawing.
- Each label must be placed in the best possible position (among all acceptable positions).

The order of preference among possible label positions varies depending on the specific application.

In the production of geographical maps, we rank label positions according to rules developed through years of experience with manual placement, which typically capture the aesthetic quality of label positions. A typical rule when labeling points (nodes) is that labels must be placed to the right and above the point. For example, in Figure 15.1(a) the number of each label position reveals the rank (priority) of the label. In addition, a point label is allowed to touch but not overlap its associated point or any other graphical feature in the drawing. In the case of labeling lines (edges), a label is allowed to touch the edge that

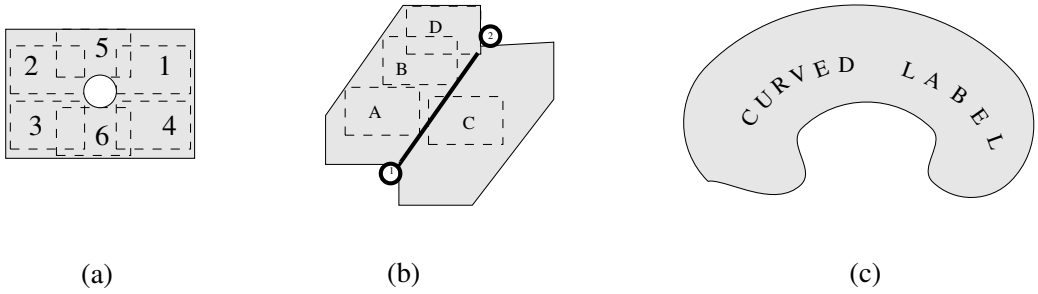

Figure 15.1 (**a**) Labeling space of a node. (**b**) Labeling space of an edge. (**c**) Labeling space of an area. Figure taken from [KT03].

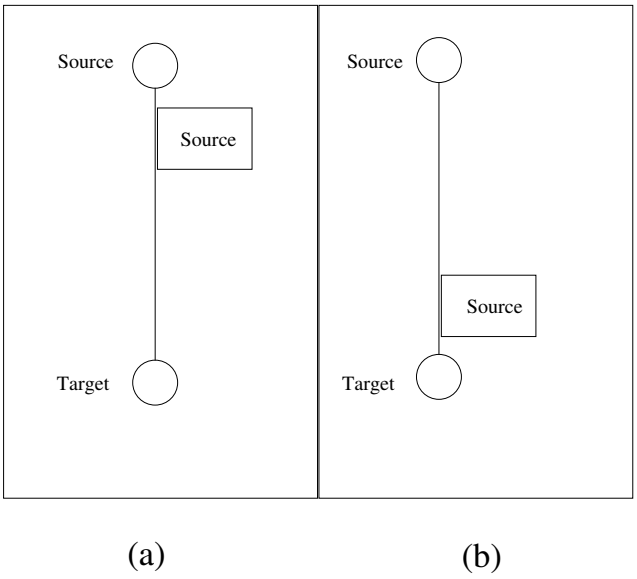

Figure 15.2 (**a**) A good label assignment. (**b**) A misleading label assignment. Figure taken from [KT03].

it belongs to, but it should not overlap any other graphical feature in a drawing. In Figure 15.1(b), where the graphical feature to be labeled is an edge, labels like A, B and D are preferable but certainly a label like C, which overlaps its associated edge, can be acceptable with some appropriate cost assigned to it. The accepted practice for placing a label associated with an area is to have the label span the entire area and conform to its shape, as shown in Figure 15.1(c). For more details on name placement rules for geographical maps, see [FA87, Imh75, vR89, Yoe72].

When the graphical objects to be labeled belong to a technical map or drawing, then, usually a different set of rules govern the preferred label positions. These rules depend on the particular application, and must follow user specifications. For example, if the graphical feature is an edge of a graph drawing, the user must be able to specify that the preferred position for an edge label is closer to the source or destination node. For example, a label of a single edge that is relevant to its source node must be placed close to the source node (see Figure 15.2(a)) to avoid ambiguity (see Figure 15.2(b)). It is important to emphasize that a user must be able to customize the rules of label quality to meet specific needs and/or

expectations. Therefore, any successful labeling algorithm must take into account the user's preferences.

15.2.2 A Definition of the Labeling Problem

Given a set F of graphical features of a map or drawing to be labeled we define the following notation:

- Λ_f is the set of all label positions for graphical feature f of F.
- Λ is the set of all label positions for all graphical features to be labeled.
- $\lambda : F \to \Lambda$ is a function that assigns a label position from Λ to graphical feature f in F, that is $\lambda(f) = \lambda_f \in \Lambda_f$.

The labeling problem can be viewed as an optimization problem where the objective is to find a label assignment of minimum total cost where each graphical feature has a label position assigned to it. Each label position λ_f that is part of a final label assignment is associated with a cost. $COST : \Lambda \to \mathcal{N}$ is a function that gives the cost of label λ_f with respect to quality.

Labeling Problem
Instance: Let F be a set of graphical features to be labeled.
Question: Find a label assignment that minimizes the following function:

$$\sum_{i \in F} \sum_{j \in \Lambda_i} COST(\lambda(i)) P(i,j)$$

Where:

$$P(i,j) = \begin{cases} 1, & if\ \lambda(i) = j, \\ 0, & otherwise \end{cases}$$

and

$$\sum_{i \in F} \sum_{j \in \Lambda_i} P(i,j) = |F|$$

Where:

$$\sum_{j \in \Lambda_i} P(i,j) = 1, \qquad i \in F.$$

□

15.3 Solving the Labeling Problem

Most of the research addressing the labeling problem has been focused on labeling graphical features of geographical and technical maps. The label placement problem is typically partitioned into three tasks: (a) labeling points (e.g., cities), (b) labeling lines (e.g., roads or rivers), and (c) labeling areas (e.g., lakes or oceans).

Progress has been made in solving the problem of assigning labels to a set of points or nodes, the *Node Label Placement* (NLP) problem [CMS95, DMM+97, FW91, Hir82, WW95, Zor90]. The problem of assigning labels to a set of lines or edges, also known as the *Edge Label Placement* (ELP) problem, has been addressed in [DKMT07, KT98, vR89, Zor90]. The general labeling problem, the *Graphical Feature Label Placement* (GFLP) problem

(where a graphical feature can be a node, edge, or area), has been addressed primarily in the context of cartography; however, it has direct application in the area of graph drawing [AF84, DF92, ECMS97, EG90, FA87, KT03].

In many practical applications, each graphical feature may have more than one label. The need for assigning multiple labels is necessary not only when objects are large or long, but also when it is necessary to display different attributes of an object. This problem is known as the *Multiple Label Placement* (MLP) problem and has been addressed in [FA87, KT06].

The labeling process is not allowed to modify the underlying geometry of geographical and technical maps which is fixed. However, one can modify a graph drawing in order to accommodate the placement of labels. In [Hu09, KT11], algorithms that modify an existing layout of a graph drawing to make room for the placement of labels are presented.

In [BDLN05, DDPP99, KM99], algorithms that combine the layout and labeling process of orthogonal drawings of graphs are presented.

An alternative approach for displaying edge labels is presented in [WMP+05]. Each edge is replaced by its corresponding edge label in the drawing. The label font starts out larger from the source node and shrinks gradually until it reaches the destination node. The tapered label also indicates the direction of the edge.

It is worth noting that both the NLP [FW91, KI88, MS91] and ELP [KT01] problems are NP-hard. Because automatic labeling is a very difficult problem we rely on heuristics to provide practical solutions for real world problems.

A variety of types of algorithms have been used in order to solve the labeling problem: greedy algorithms [CMS95, Hir82], exhaustive search algorithms [DF92, EG90, FA87], algorithms that simulate physical models (i.e., Simulated Annealing [CMS95]), algorithms that reduce the labeling problem to a variant of 0-1 integer programming [Zor90], algorithms that restrict the labeling problem to a variant of the 2-SAT problem [FW91, WW95], and algorithms that transform the labeling problem into a matching problem [KT98, KT03, KT06].

15.3.1 The GFLP Problem

Most labeling algorithms that address the general labeling problem are based on local and exhaustive search algorithms [DF92, EG90, FA87]. These algorithms perform well for small problems. These methods use inferior optimization techniques, as pointed out in [Zor90] and verified in [CMS95]. Actually, these methods use a rule based approach to evaluate good label placement and variants of depth-first search to explore different labeling configurations. The approach in [ECMS97] uses simulated annealing to find solutions for the general labeling problem, and it separates the cartographic knowledge needed to recognize the best label positions from the optimization procedure needed to find them.

All of the above techniques for the general labeling problem first create an initial label assignment in which conflicts between labels are allowed. Then conflicts are resolved by repositioning assigned labels until all conflicts are resolved, or no further improvement can be achieved. Furthermore, they start with a rather small initial set of potential label positions from which they derive a final label assignment. The performance of these techniques decreases when the number of potential label positions increases.

In [KT03] the labeling problem is transformed into a matching problem. The general framework of this technique is flexible and can be adjusted for particular labeling requirements. In the next section this technique is presented in more detail.

A practical matching algorithm for the GFLP problem

The placement of labels is a post-layout operation (i.e., performed on a fixed geometry of nodes and edges). The basic idea behind this labeling technique is the following: a set of discrete potential label solutions for each object is carefully selected. This set of labels is reduced by removing heavily overlapping labels. Finally, an assignment of labels is performed by solving a variant of the matching problem. This method is shown in Figure 15.3. An example of the resulting label placement is given in Figure 15.4.

Basic Labeling Algorithm

INPUT: A drawing Γ and a set F of objects to be labeled.
OUTPUT: A label assignment free of overlaps.
1. A set of discrete potential label solutions for each object in F is carefully selected.
2. This set of labels is reduced by removing heavily overlapping labels. The remaining labels are assigned to groups, such that, if two labels overlap then they belong to the same group.
3. Labels are assigned by solving a variant of the matching problem, where at most one label position from each group is part of the solution.

Figure 15.3 Basic labeling algorithm.

Next, the three basic steps of the basic labeling algorithm are presented in detail.

Selecting labels

To find a set of discrete label positions for each graphical feature, a number of heuristics can be used. For points, a number of label positions that touch their corresponding point is defined. In most algorithms a finite set of potential label positions are associated with each point, typically the size of this set is four or eight as shown in Figure 15.5 (see also [CMS95]).

It is generally accepted, especially in the framework of cartography, that area labels must follow the general shape of their corresponding area, and that they must be inside the boundaries of the area. For each area, a number of potential label positions is defined according to the techniques described in [FA87, Fre88, PF96, vR89].

Next, a simple heuristic for finding a set of label positions corresponding to edges of graph drawings is presented. As Figure 15.6 illustrates, a number of equally spaced points on each edge is defined. Each assigned label position λ_i is associated with exactly one of these points i, such that one of the corners of label λ_i coincides with point i. In addition, label λ_i does not overlap its corresponding graphical feature or any other graphical feature (except other label positions). A global approach for finding an initial set of label positions for non-horizontal edges can be found in Section 15.3.2 and in [KT98].

Reducing labels

The size of the initial set of label positions must be kept reasonably small since it affects the performance of any labeling algorithm.

In order to reduce the set of label positions an intersection graph is first created, where each label position is a node and if two label positions intersect then there is an edge

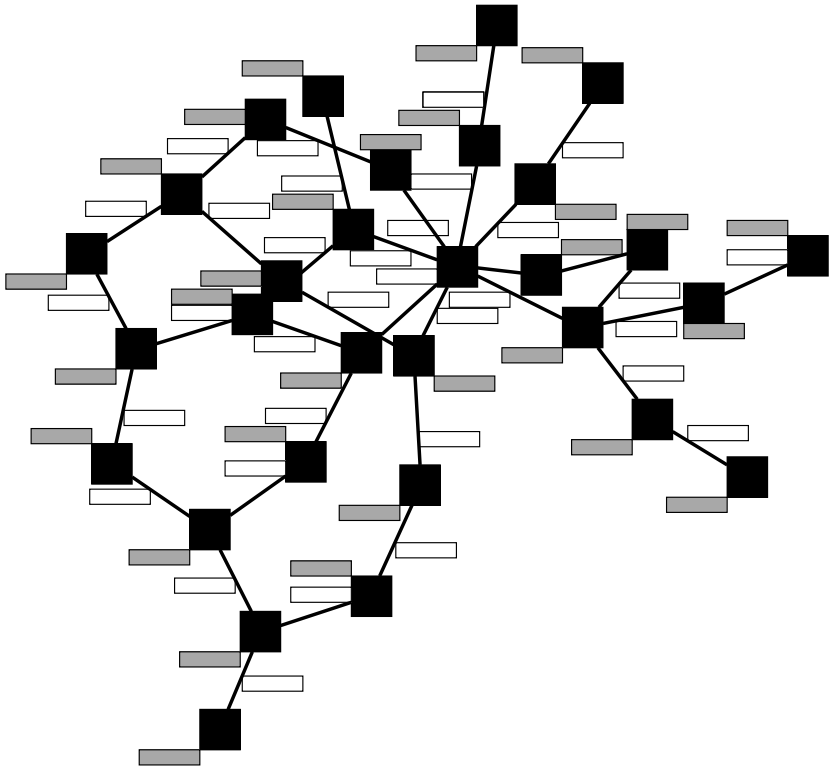

Figure 15.4 A force-directed drawing where labels are positioned by the matching technique for the GFLP problem. The labels are parallel to the horizontal axis. The grey boxes are node labels and the white boxes are edge labels. Figure taken from [KT03].

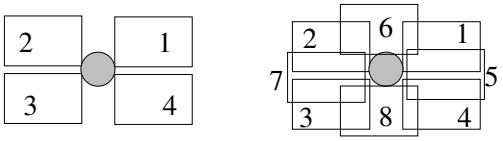

Figure 15.5 Potential label positions for a point. Figure taken from [KT03].

connecting their corresponding nodes. If label positions are parallel to the axis then overlaps can be detected using the techniques for detecting overlaps among isothetic rectangles in $O(n \log n + K)$ time (n is the number of rectangles and K the number of intersections) [Ede83a, Ede83b]. Otherwise, in order to detect overlaps of labels with arbitrary orientation, the techniques of [GJS96] can be used to detect intersections between convex polygons in $O(n^{4/3+\epsilon} + K)$ time (n is the total number of vertices of the polygons, K is the number of pairs of polygons that intersect, and ϵ is any constant greater than zero).

Then, heavily overlapping labels are removed. The remaining labels are assigned to groups, such that, if two labels overlap then they belong to the same group. The goal of the third step of the algorithm is to select at most one label from each group as part of the solution. This way, the algorithm will produce a label assignment free of overlaps.

The optimal solution would be one with the maximum number of minimum size complete subgraphs (groups) of the intersection graph, with the additional constraint that each object has a large number of label positions as part of some groups. It is most likely to have a

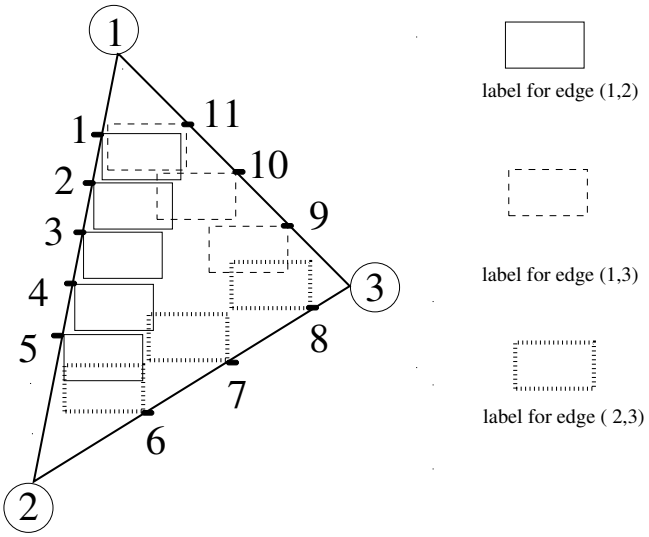

Figure 15.6 A graph drawing with label positions assigned to each edge of the drawing. Figure taken from [KT03].

successful label assignment when each object has a large number of potential label positions associated with it. In reality the goal is to find an independent set of complete subgraphs of label positions. This can be done by using heuristics based on techniques that solve the independent set problem.

Heavily overlapping labels are removed, while maintaining a large number of potential labels for each object f by keeping track of the number of labels associated with f. Our aim is to reduce the intersection graph into a set of disconnected subgraphs.

First, in order to make this process more efficient a preprocessing step is applied that eliminates unnecessary labels or assigns labels in obvious cases. For example, if a label position l of an object f is free of overlaps, then all label positions for f with lower ranking than l can be safely removed.

Next, an appropriate number of overlapping labels is removed. A simple and very successful (according to experiments) technique for removing overlapping labels is the following: If a subgraph c must be split, then the node with the highest degree is removed from c, unless that node corresponds to a label position of some object with very few label positions. In that case the next highest degree node from c is removed. This process is repeated until either c is split into at least two disjoint subgraphs, or c is complete.

Matching labels to objects

To further clarify the main idea of this technique the *matching* graph is introduced:

DEFINITION 15.1 Given a drawing Γ, a set F of graphical features to be labeled, and a set Λ of label positions for F, the matching graph $G_m(V_f, V_c, E_m)$ is defined as follows:

- Each node in V_f corresponds to a graphical feature in F.
- Each node in V_c corresponds to a group of overlapping labels.
- Each edge (i, j) in E_m connects a node i in V_f, to a node j in V_c, if and only if the graphical feature that corresponds to i has a label position that is a member of the group that corresponds to j.

Notice that G_m is a bipartite graph and the cost of assigning label l to graphical feature f is the weight of edge (f, l) in G_m. Therefore, a maximum cardinality minimum weight matching for graph G_m will give us an optimal (maximum number of labels with minimum cost) label assignment with no overlaps with respect to the *reduced* set of label positions.

By representing the labeling problem as a bipartite graph, the inherent hardness of the problem is revealed. According to [KR92], the labeling problem is closely related to the *independent set* problem. Indeed, consider the very simple case where there is only one potential label position for each graphical feature in the drawing, the problem of assigning labels to the maximum number of graphical features is equivalent to finding a maximum size independent set.

Once the set of groups is found, the construction of the matching graph is trivial. A final label assignment can be found by solving the maximum cardinality minimum weight matching problem (see [GK95, Tar83] for efficient algorithms) for graph G_m. The size of the matching graph depends not only on the size of the input drawing, but also on the size of set Λ of labels and the density of overlaps. Notice that at most one label position from each group may be part of a label assignment. Thus, a matching of graph G_m produces an assignment free of overlaps. Because the label assignment is free of overlaps, the cost of each label position will depend only on the ranking of that label. This implies that the cost of each label position can be computed by a preprocessing step.

15.3.2 The ELP Problem

The problem of assigning labels to a set of lines or edges, also known as the *Edge Label Placement* (ELP) problem, has been addressed in the context of geographical and technical maps [AH95, ECMS97, vR89, WKvK$^+$00, Zor90] and graph drawing [KT98, DKMT07]. Furthermore, any of the techniques for solving the general labeling problem (see section 15.3.1) can be applied in solving the ELP problem.

In the context of geographical and technical maps edges are linear features. Labels should be placed alongside and parallel to rivers, boundaries, roads or linear features. If the linear feature is curved, the shape of the label must follow the curvature of the linear feature. The positioning of linear labels has the greatest degree of freedom since labels can be placed almost anywhere along the linear feature, thus cartographers have been focusing their efforts on finding the right shape of the linear label.

However, in the context of graph drawing, placing labels to edges is a more complicated process. Edges are not necessarily long, they are usually straight lines or polygonal chains and they have to follow user preferences and specifications. For example an edge label might be related to the source node of the edge, thus it must be placed closer to the source node rather than the target node to avoid a misleading label assignment (see Figure 15.2).

In [DKMT07] a labeling system is presented that includes a very functional interface and labeling engine that addresses the ELP problem in the context of a graph drawing editor. It is noteworthy that the interface of that system allows the user to set the labeling preferences interactively.

In the following section a fast and simple technique, first proposed in [KT98], is presented for solving the problem of positioning text or symbol labels corresponding to edges of a graph drawing.

A fast and simple algorithm for labeling edges of graph drawings

This technique is based on the matching technique for solving the general labeling problem presented in section 15.3.1.

The technique works for labels that are parallel to the horizontal axis, and have approximately equal height and arbitrary width. In order to simplify the discussion the following assumptions are made:

- All labels have the same size.
- Each edge has only one label associated with it.

The goal of this technique is to assign to each edge a label position that is free of overlaps and touches only its associated edge. The main idea of this technique is the following:

First, a set Λ of label positions is produced. Next, label positions are grouped such that each label position that is part of a group overlaps every other label position that belongs to the same group. Then, edges to label positions are matched by allowing at most one label position from each group to be part of a label assignment by using a fast matching heuristic. The key to restricting the ELP problem to a matching problem is to create a suitable initial set of label positions.

The initial set of label positions is created in the following way. The input drawing is divided into consecutive horizontal strips of equal height. The height of each strip is equal to the height of the labels. Next, a set of label positions Λ_e for each edge e is found. Each label position must be inside a horizontal strip. Labels are slided inside each horizontal strip until a label touches its edge, say e. That label position is included into set Λ_e if it does not overlap any other graphical feature or only overlaps label positions of some edge other than e, as shown in Figure 15.7. Label positions that overlap nodes or edges of the layout are not considered. Also label positions are not allowed to intersect their associated edges. Label positions lie entirely inside horizontal strips. Thus, label positions can only overlap other labels that belong to the same horizontal strip. Hence, the following are true:

- A label position of an edge e does not overlap any other label position of e.
- If two label positions overlap then they are inside the same horizontal strip.
- Each label position overlaps at most one other label position.

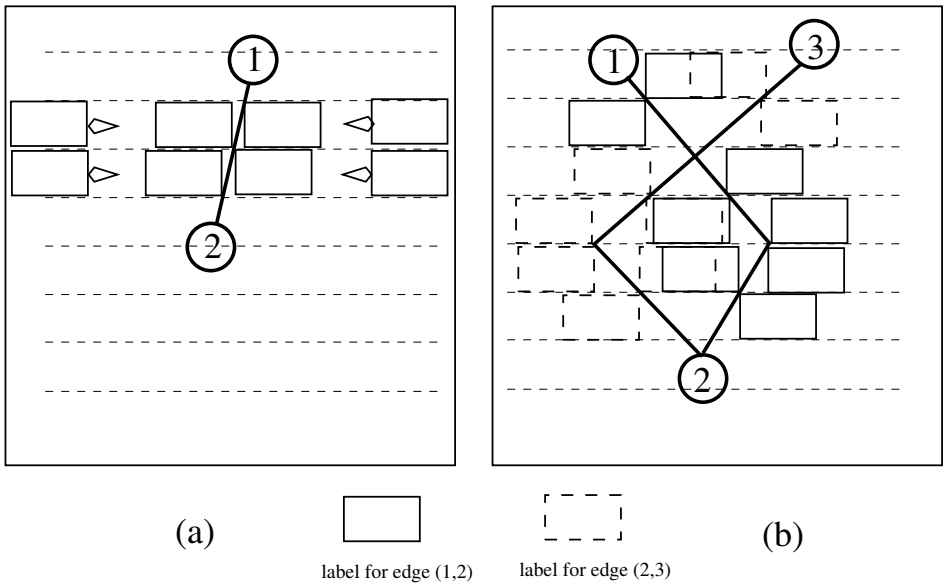

Figure 15.7 Assigning potential labels to edges of a drawing. Figure taken from [KT98].

If two label positions overlap then they belong to the same group. If a label position is free of overlaps then it belongs to a single member group.

The size of the initial set of label positions must be kept reasonably small since it affects the performance of any labeling algorithm. The above method of defining a set of potential label positions is very practical and effective because it partitions the solution space and identifies the areas of the drawing where conflicts of label assignment may occur. In addition, it significantly reduces the search space for potential conflicts (overlaps).

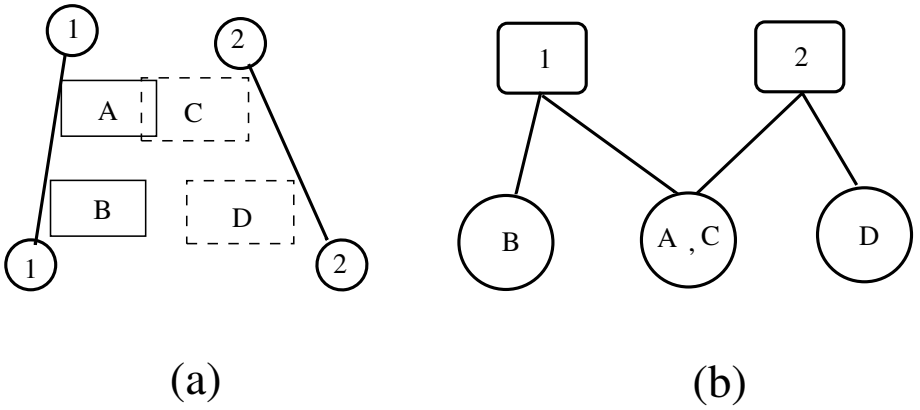

Figure 15.8 (**a**) A simple drawing with label positions for each edge. (**b**) The corresponding matching graph. Figure taken from [KT03].

In Figure 15.8 a simple example of how to construct the matching graph (see Def. 15.1) is presented. Figure 15.8(a) represents a simple drawing with two edges and two label positions for each edge. Figure 15.8(b) shows its corresponding matching graph. Label positions that overlap belong to the same node in the matching graph, in this example label position A of edge 1 overlaps label position C of edge 2, thus they are represented by a single node in the matching graph.

Since at most one label position from each group may be part of a label assignment, a matching of graph G_m produces an assignment free of overlaps. A maximum cardinality matching of graph G_m assigns labels to the maximum number of edges.

A description of the labeling technique is given in the algorithm of Figure 15.9.

Algorithm ELP
INPUT: A drawing Γ of graph $G(V, E)$.
OUTPUT: A label assignment free of overlaps.
1. Split Γ into horizontal strips.
2. Find all label positions for each edge and construct the groups of overlapping labels.
3. Create the matching graph G_m for Γ.
4. Match label positions to edges, by finding a maximum cardinality minimum weight matching of G_m.

Figure 15.9 Algorithm ELP.

The size of the matching graph depends on the size of set Λ of label positions. Unfortunately the size of Λ can be large with respect to the size of the original graph G. This implies that a typical matching algorithm might take a long time. The best algorithms for finding a maximum cardinality minimum weight matching of G_m take more than quadratic time with respect to the size of G_m [GK95, Tar83]. In order to reduce the time complexity of the matching, in the next section a heuristic is presented that finds a maximum cardinality matching with low total weight in linear time with respect to the size of G_m, by taking advantage of the structure and properties of graph G_m.

A Fast Matching Heuristic

Here, a fast heuristic is presented that solves the maximum cardinality matching problem for a matching graph where each node corresponding to a group of overlapping labels has degree at most two. The fast heuristic that solves the matching problem takes advantage of the simple structure of the matching graph. By construction, each node in V_c has degree at most 2 (see Figures 15.7 and 15.8). The algorithm of Figure 15.10 finds a maximum cardinality matching for G_m.

Algorithm Fast Matching
INPUT: Matching graph G_m.
OUTPUT: A maximum cardinality matching for G_m with low total weight.
1. *If* the minimum weight incident edge of a node in V_f connects this node to a node in V_c of degree 1 *then*:
  1.1. Assign this edge as a matched edge.
  1.2. Update G_m.
2. *If* a node in V_f has degree 1 *then*:
  2.1. Assign its incident edge as a matched edge.
  2.2. Update graph G_m.
3. *Repeat* Steps 1 and 2 until no new edge can be matched.
4. Delete all nodes of degree 0 from G_m.
5. *For* each node f in V_f *do*
  5.1. Remove all but the two incident edges of f with the least weight.
6. The remaining graph consists of simple cycles and/or paths.
  6.1. Find the only two maximum cardinality matchings for each component.
  6.2. Choose the matching of minimum weight.

Figure 15.10 Algorithm Fast Matching.

Note: The *Update* G_m operation removes the two nodes incident to a new matched edge and stores that edge and its incident nodes as part of the matching. Also removes all incident edges from the two nodes.

In Step 1 matched edges are found that are part of any optimal solution. In Step 2 edges are matched to those nodes in V_f that are of degree 1. If two nodes of degree 1 in V_f are connected to the same node in V_c, as matched edge is chosen the edge with minimum weight. This implies that one of the edges will have no label. In Step 4 nodes are removed from G_m that correspond to either edges that have no potential labels assigned to them or have potential labels that will not be part of a final labeling assignment. In Step 5, for each

node in V_f of degree more than 2, only its two incident edges of least weight are kept and the rest of the edges are removed. The remaining bipartite graph has a simple structure: It consists of simple cycles or simple paths, because each node in V_f has degree 2 and each node in V_c has degree at most 2. Each path or cycle has exactly two maximum cardinality matchings. It is trivial to find both of them by simply traversing the cycle or path and picking as part of the matching only the even or odd numbered edges.

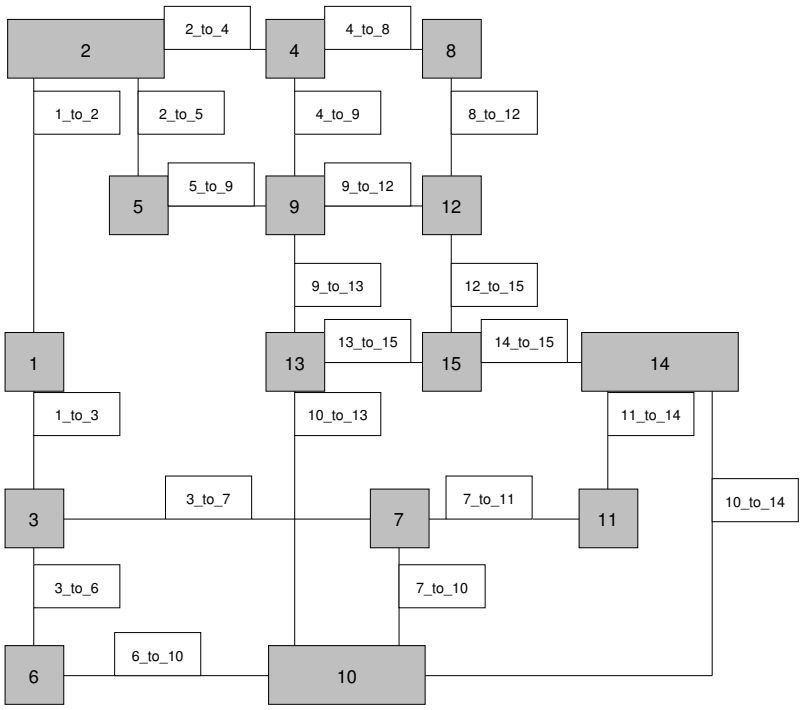

Figure 15.11 An orthogonal drawing with edge labels produced by the fast ELP technique. Figure taken from [KT03].

It is trivial to see that Algorithm Fast Matching runs in linear time. Notice that it also finds a maximum cardinality matching with low total weight because in the last step it considers only the two incident edges of nodes in V_e with the lowest weight. Figure 15.11 shows a label assignment produced by the fast ELP technique.

Further Improvements

For the above fast and simple technique for labeling edges of graph drawings it is clear that the longer and the more vertical the edges are, the more potential label positions are associated with each edge. Thus, the greater the possibility for the labeling algorithm to assign a label to each of these edges. Therefore, hierarchical drawings are particularly suitable for this algorithm since edges are usually long and almost vertical. This technique performs very well also for straight-line drawings, such as ones produced by force-directed and circular techniques. One weakness of this labeling technique is that it ignores horizontal

edges or edge segments. Thus, as presented, this technique is not suitable for orthogonal drawings. However, one can use the general technique by dividing an orthogonal drawing into horizontal and vertical strips in order to find a set of label positions, followed by the assignment of labels to edges. Figure 15.12 shows the results with an example.

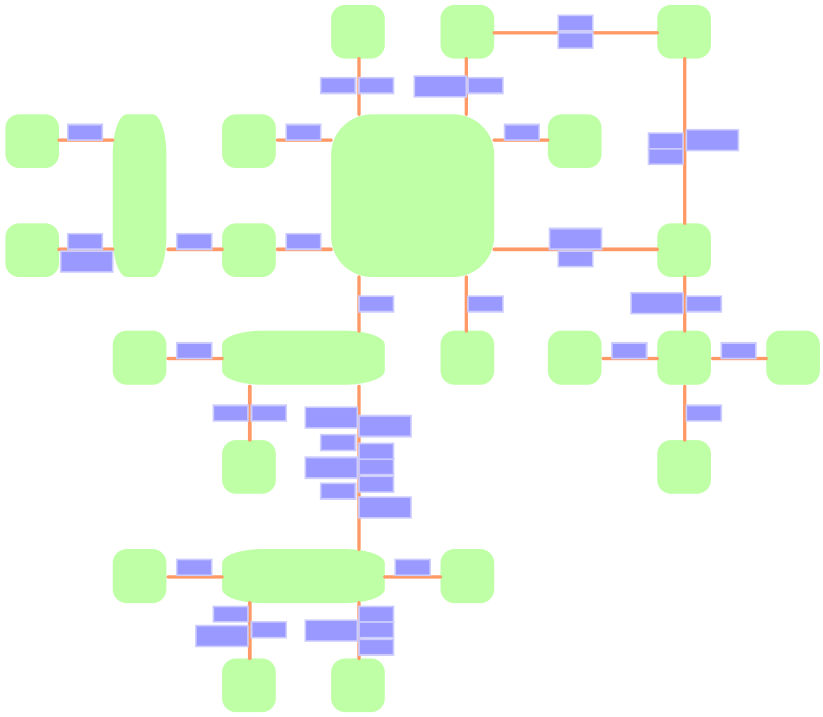

Figure 15.12 An orthogonal drawing with edge labels, which contains many horizontal edge segments, produced by the fast ELP technique. Figure taken from [DKMT07].

In addition, when drawings are very dense or there is a large number of oversized labels, the default label assignment produced by the labeling system might not be satisfactory. In such cases, the user can fine tune the algorithm by relaxing the labeling quality constraints by allowing overlaps (see Figure 15.13).

15.3.3 The NLP Problem

The problem of assigning labels to a set of points or nodes, also known as the *Node Label Placement* (NLP) problem, has been extensively studied in the context of automated cartography and many successful algorithmic approaches have been introduced [CMS95, DMM+97, FW91, Hir82, WW95, Zor90]. Also, any of the techniques for solving the general labeling problem (see section 15.3.1) can be applied to the NLP problem.

Algorithms based on local and exhaustive search [DF92, EG90, FA87] and simulated annealing [ECMS97] are well suited for solving the NLP problem. Experimental results [CMS95] have shown that simulated annealing outperforms all algorithms based on local and exhaustive search. In addition simulated annealing is one of the easiest algorithms to implement.

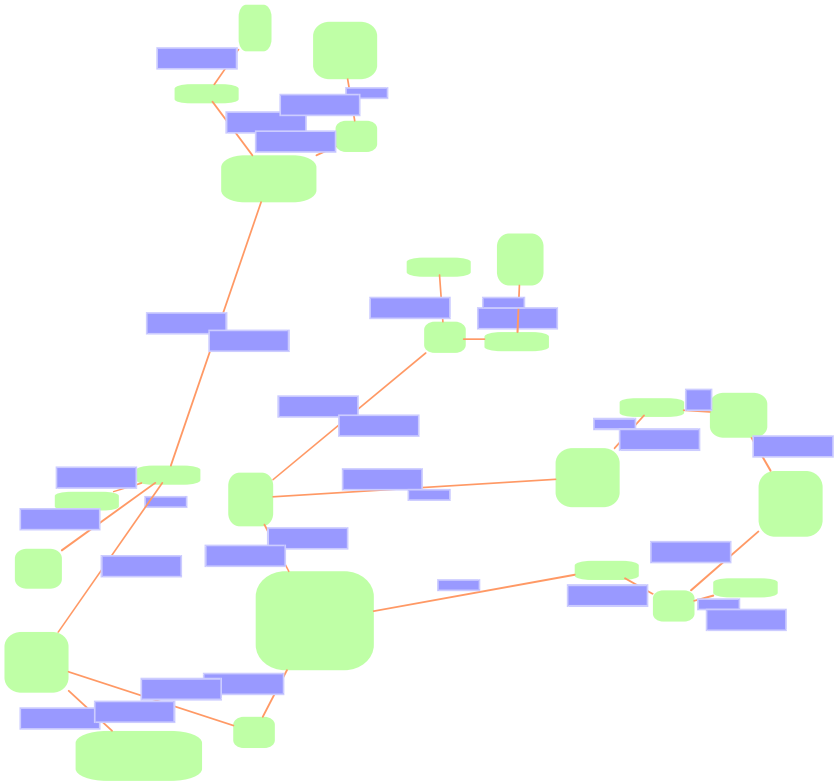

Figure 15.13 A circular drawing with edge labels, where labels are allowed to overlap other graph objects, produced by the fast ELP technique. Figure taken from [DKMT07].

These algorithms start with a rather small initial set of potential label positions from which they derive a final label assignment. This is because the size of the initial set of label positions plays a critical role in the performance of these algorithms. This precondition works well when solving the NLP problem. For example, each point is given at most four or eight potential label positions (see Figure 15.5).

Approximation algorithms for restricted versions of the NLP problem are presented in [DMM$^+$97, FW91]. Specifically, the approach of [FW91] assigns labels of *equal* size to all points while attempting to maximize the size of the assigned labels. The work in [WW95] improves the results in [FW91] by using heuristics. A similar approach has been taken in [DMM$^+$97]. In effect, finding the maximum label size is equivalent to finding the smallest factor by which the map has to be zoomed out such that each point has a label assigned to it. However, it is not clear how these techniques can be modified to solve real-world problems, including the labeling of graphical features of graph drawing, where the label size is usually predefined and labels are not necessarily of equal size.

Another approach to solve the NLP problem is based on the sliding model, where sliding labels can be attached to the point they label anywhere on their boundary. This model was first introduced in [Hir82] who gave an iterative algorithm that uses repelling forces between labels in order to eventually find a placement of labels. A polynomial-time approximation scheme and a fast factor-2 approximation algorithm for maximizing the number of points that are labeled by axis-parallel sliding rectangular labels of common height, based on the sliding model, is presented in [vKSW99].

15.3.4 The MLP Problem

Many algorithms exist for the labeling problem; however, very little work has been directed toward positioning many labels per graphical feature in a map or drawing [FA87, Fre88, KT06]. This problem is known as the *Multiple Label Placement* (MLP) problem.

In existing automated name placement systems for geographic maps simple techniques have been utilized to address the MLP problem [FA87, Fre88]. Specifically, each feature to be labeled is partitioned into as many pieces as the number of labels for that feature. Then, labeling algorithms for single label per graphical feature may be applied to the new set of partitioned graphical features. In many applications, this straightforward approach presents some difficulties. For example, it might be necessary or preferable to position labels that are associated with the same graphical feature next to each other (e.g., two labels assigned to an edge must be close to the source node of the edge). This is often the case when labels describe more than one attribute of the same feature. Furthermore, the feature to be labeled might be a point or an area. Then, we must partition the solution space and assign one label to each of the partitions. However, efficiently partitioning the solution space is as hard as solving the original labeling problem. Even when we need to place more than one label associated with a linear graphical feature in regular intervals from each other, this approach seems weak. Since, by splitting the features beforehand we eliminate solution space that otherwise could be used to position a label.

One can avoid the situations described in the previous paragraph by allowing each of the labels to be placed in any legible label position of the associated graphical feature. An iterative approach based on existing labeling algorithms that assigns one label per graphical feature can be used to produce a solution. This can be done by applying these algorithms as many times as the number of labels per graphical feature. This scheme presents a new challenge: most labeling algorithms are based on local and exhaustive search. Thus, their performance (running time and quality of solutions) is sensitive to the size of the graphical features to be labeled and to the density of the drawing. Clearly, if each graphical feature in a drawing is associated with i labels, then the size of the problem is i times larger. Therefore, the above techniques might be slow even for small instances.

In [KT06] the MLP problem is treated in the context of graph drawing. A framework for evaluating the quality of label positions is presented. In addition, two algorithmic schemes are presented: (i)A simple and practical iterative technique and (ii) A flow-based technique which is an extension of the matching technique presented in section 15.3.1. In the following sections these techniques will be presented in detail.

Labeling quality rules for the MLP problem

Multiple labels per graphical feature are needed not only when objects are very long (i.e., long edges) and repetition is necessary, but also when more than one attribute per graph object must be displayed. Therefore, some additional considerations have to be taken into account with respect to the quality of a label assignment, when graphical features have many labels. Specifically, we must take into account how labels for the same graphical feature influence each other. For example, many times each of the labels corresponds to some attribute of a graphical feature and the relative position of a label with respect to other labels of the same graphical feature reveals that attribute.

Next, we present some constraints that may be used to ensure that each label is unambiguous, easily read and recognized, when more than one label is associated with a graphical feature. These constraints can be divided into three general categories: (i) *proximity*, (ii) *partial order*, and (iii) *priority*. In order to illustrate the three different sets of constraints we will use as an example the labeling of a single edge (s, t) with two labels l_s and l_t. Label

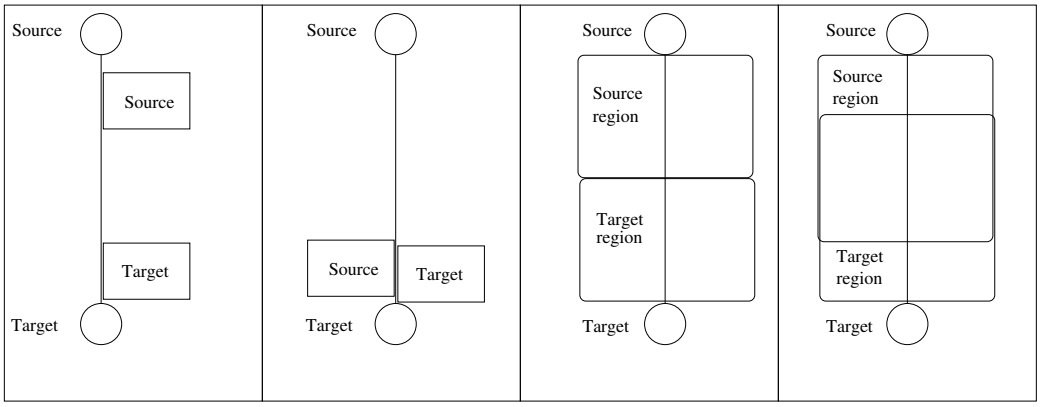

Figure 15.14 (**a**) A preferable label assignment. (**b**) A misleading label assignment. (**c**) Defining strict proximity constraints. (**d**) Defining relaxed proximity constraints. Figure taken from [KT06].

l_s is associated with the source node and label l_t is associated with the target node, as shown in Figure15.14(a).

Proximity:

Label l_s (resp. l_t) must be in close proximity with the source (resp. target) node to avoid ambiguity. Therefore, it is necessary to define a maximum distance from the source (resp. target) node that label l_s (resp. l_t) may be positioned. When edge (s, t) is associated with exactly one label, then that label may be located anywhere inside the solution space. If there are more than one label associated with (s, t), then each label must be positioned inside an area that is a subset of the solution space.

In Figure 15.14 we illustrate the importance of the proximity constraints. For example the label assignment in Figure 15.14(a) is a preferable assignment. The assignment in Figure 15.14(b) does not convey clearly the meaning of the labels, because they are very close to the target node; hence by observing the picture we cannot establish with certainty that the source label is associated with the source node. In Figure 15.14(c) the proximity constraint is that the distance between the source (resp. target) node and its label must be at most half the length of the edge. This implies that the source (resp. target) label must be inside the source (resp. target) region. The defined proximity constraints in Figure 15.14(c) are too restrictive, since the defined regions do not intersect. One could define more relaxed proximity constraints, as shown in Figure 15.14(d), where intersecting of different regions is allowed. In practice the latter is preferable since it increases the labeling solution space and improves the possibility for finding a labeling assignment, especially in cases where the drawing is crowded.

Partial Order:

A label associated with the source (resp. target) node must be closer to the source (resp. target) node than any other label to avoid ambiguity. Thus, in many cases, it is appropriate to define a partial order between labels of the same graphical feature according to some invariant (e.g., x or y axis, distance from a fixed point).

In Figure 15.15(c) we present an example where the absence of a partial order rule produces a misleading label assignment, since by simply looking at the picture we associate the target (resp. source) label to the source (resp. target) node. In Figures 15.15(a) and

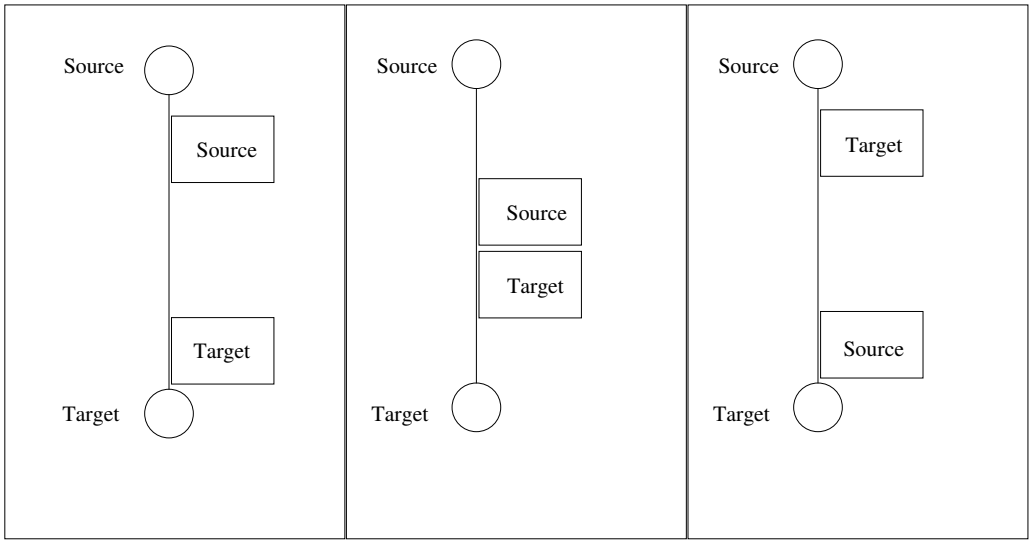

Figure 15.15 (**a**) A preferable label assignment. (**b**) An acceptable label assignment. (**c**) A misleading label assignment. Figure taken from [KT06].

15.15(b) the additional condition that a label associated with the source node must be closer to the source node than the label associated with the target node ensures the correct interpretation of the label assignment. If we define restrictive proximity constraints, as shown in Figure 15.14(c), then a partial order constraint is not necessary. However, if we relax the proximity constraints, as shown in Figure 15.14(d), then we need to define a partial order constraint in order to avoid misleading labeling assignments.

Priority:

In many cases, it is impossible to assign all labels associated with a graphical feature, due to the density of the drawing. Then, the user might prefer to have the important labels assigned first, and then assign the rest of the labels if there is available space.

These three sets of constraints present a succinct framework for a good label assignment with respect to the MLP problem.

In the following sections we focus on two sets of heuristics, iterative and flow-based, to solve the MLP problem.

An iterative algorithm for the MLP problem

First a simple iterative approach to solve the problem of assigning multiple labels to each graphical feature of a drawing is presented. For simplicity, let us assume that each graphical feature is associated with the same number of label positions. The main idea is the following: existing algorithms solve the labeling problem for single label per graphical feature. Therefore, one could solve the MLP problem by applying these algorithms as many times as the number of labels per graphical feature. This method consists of a main loop, and we execute the loop as many times as the number of labels per graphical feature. In particular, at the *i-th* execution of the loop, we assign the *i-th* label to each graphical feature.

This technique can take into account all three sets of constraints: (a) proximity (by considering only the label positions that respect the proximity rules), (b) partial order

(by eliminating from the set of potential label positions, after each execution of the loop, the label positions that do not respect the partial order) and (c) priority (by selecting, if possible, the label position of highest priority among the available label positions). One can refine this technique by first finding a set of label positions before entering the loop, and then executing inside the loop only the step of positioning labels. The refinement works because the cited labeling algorithms produce a label assignment from an initial finite set of discrete potential label positions. The refined algorithm is shown in Figure 15.16.

Iterative Algorithm
INPUT: A drawing Γ, a set of graphical features F in Γ to be labeled,
a number N of labels for each graphical feature f in F.
OUTPUT: A label assignment.
1. Find an initial set of label positions L.
2. *For* $i = 1$ to N *do*:
2.1. Assign the *i-th* label to each graphical feature in F from the set L of potential labels using existing labeling algorithms.
2.2. Remove potential label positions from L that overlap already assigned labels.

Figure 15.16 Iterative algorithm.

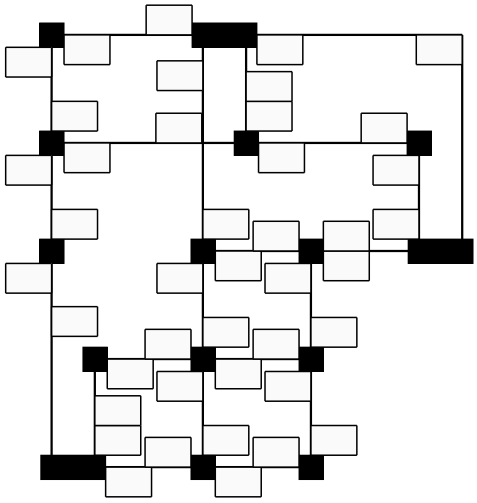

Figure 15.17 An orthogonal drawing with two labels per edge, positioned by the Iterative algorithm. Figure taken from [KT06].

Even though this technique is very attractive, especially because it can be realized by using existing labeling algorithms, it presents some challenges that have to be addressed. Labeling techniques based on local or exhaustive search first create an initial label assign-

ment where conflicts between labels are allowed. Then conflicts are resolved by repositioning assigned labels until all conflicts are resolved, or no further improvement can be achieved. When applying these techniques in the context of the iterative algorithm one can either apply repositioning only for labels assigned in the current run of the loop or for any assigned label (even in previous runs of the loop). In either case such techniques are slow.

This iterative approach is especially suited for the labeling algorithms presented in [KT98, KT03], because they first find a set of label positions, and then they produce a label assignment in a single step without any repositioning of labels (see Figures 15.17 and 15.18).

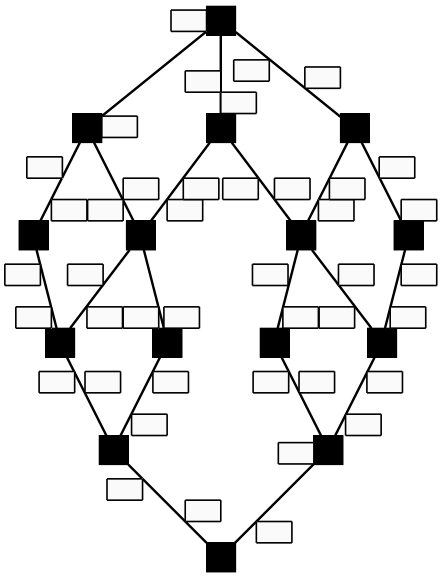

Figure 15.18 A hierarchical drawing with two labels per edge, positioned by the Iterative algorithm. Figure taken from [KT06].

A Flow-Based algorithm for the MLP problem

The matching technique presented in Section 15.3.1 can be further extended to support placement of more than one label per graphical feature of a graph drawing. The algorithm presented here assigns label positions in a non-iterative fashion. It solves the MLP problem by reducing it to an assignment problem.

First the matching graph G_m is created (see Section 15.3.1 for more details).

Next, the matching graph G_m is transformed into a flow graph $G_{flow}(s, t, V_f, V_c, E_f)$. G_m is converted to an st-graph by introducing two nodes s and t. Node s is connected to each node in V_f, and node t is connected to each node in V_c, as shown in Figure 15.19.

Finally capacities to each edge of the flow graph G_{flow} are assigned in the following way:

- Each edge of the original matching graph has capacity one.
- Each edge (c, t) of G_{flow} incident to the target node has capacity one.
- Each edge (s, v) incident to the source node has capacity equal to the number of labels associated with the graphical feature of the input graph that is represented by node v in G_m.

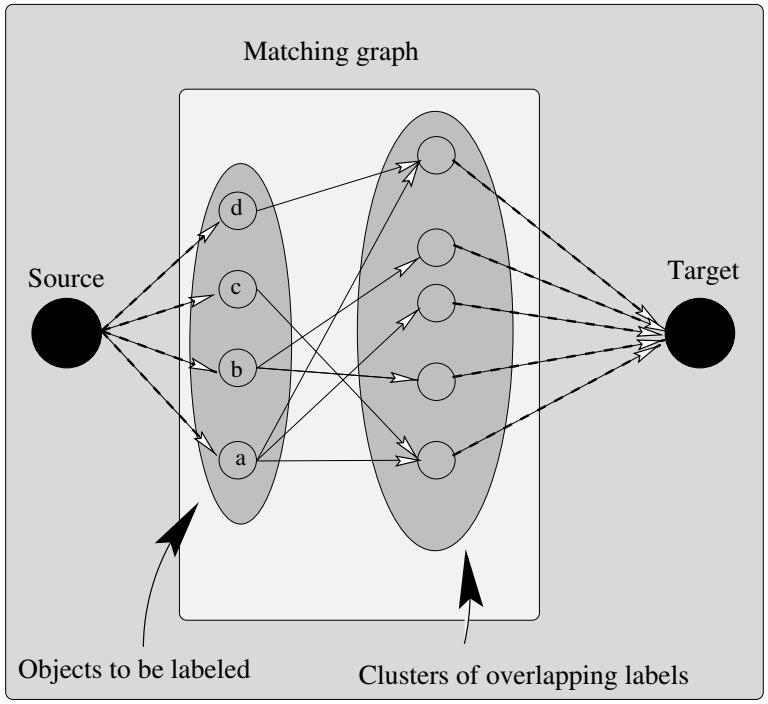

Figure 15.19 The flow graph. Figure taken from [KT06].

Clearly a maximum flow of graph G_{flow} will produce a maximum cardinality label assignment with respect to the set of labels encoded in the matching graph. Sophisticated techniques can solve the maximum flow problem in $O(nm \log n)$ time [AMO93], where n is the number of vertices and m is the number of edges of the flow graph.

This technique is summarized in the algorithm of Figure 15.20.

Flow-based Algorithm
INPUT: A drawing Γ, a set of graphical features F in Γ to be labeled,
a number $M(f)$ of labels for each graphical feature f in F.
OUTPUT: A label assignment free of overlaps.
1. Find a set of label positions for each graphical feature in the drawing.
2. Arrange overlapping label positions into groups.
3. Create the matching graph G_m.
4. Augment graph G_m to a flow graph G_{flow}.
5. Assign capacities to each edge of G_{flow}.
6. Assign cost to edges of G_{flow}.
7. Find the maximum flow minimum cost of graph G_{flow}.
8. Assign labels according to the results of Step 7.

Figure 15.20 Flow-based algorithm.

The two most time consuming steps of the above algorithm are the detection of overlaps between label positions and the matching produced by running a maximum flow minimum cost algorithm on the flow graph. Clearly the time required for those two steps depends highly on the size of the initial set of label positions. Therefore, the performance of the above algorithm is closely related to the size of the initial set of label positions.

One point that needs to be emphasized is that the framework just described can take into account the cost of a label assignment with respect to priority, proximity and aesthetic criteria. Since the final label assignment is free of overlaps, one may assume that there is no cost associated with the relative position of any pair of assigned labels. Each edge in the bipartite graph G_m connects a graphical feature to a label position of that feature that belongs to some group. The cost of label position l of graphical feature f is included as the weight of edge (f, l) in the matching graph. Then, by assigning to edges incident to source and target nodes weight equal to zero, one can find a maximum cardinality minimum cost label assignment for the reduced MLP problem by solving the maximum flow minimum cost problem for the flow graph G_{flow} (see [AMO93] for efficient techniques for solving the flow problem). Figures 15.21 and 15.22 show label assignments produced by the Flow-based algorithm.

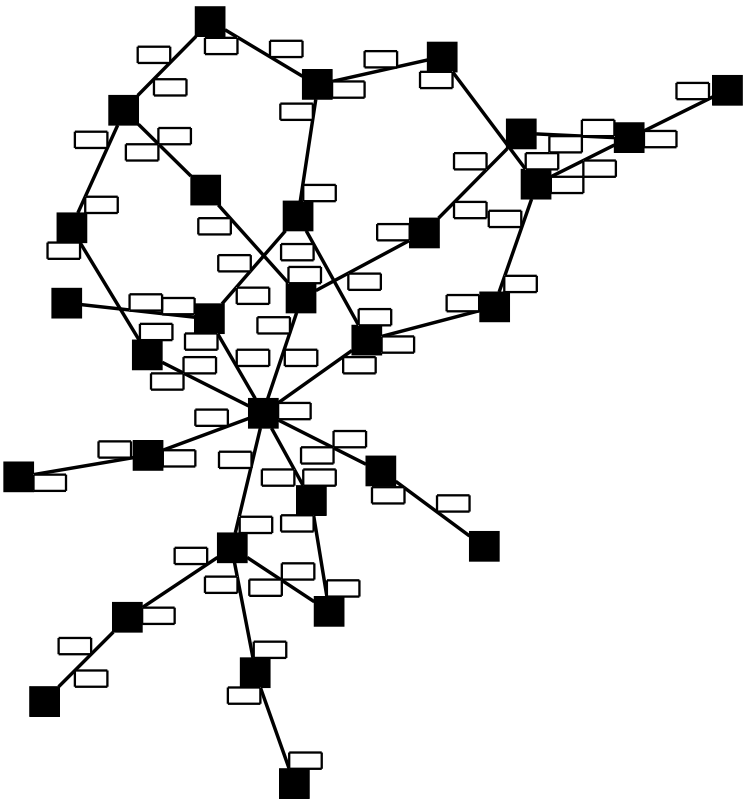

Figure 15.21 A force-directed drawing with two labels per edge, positioned by the Flow-based algorithm. Figure taken from [KT06].

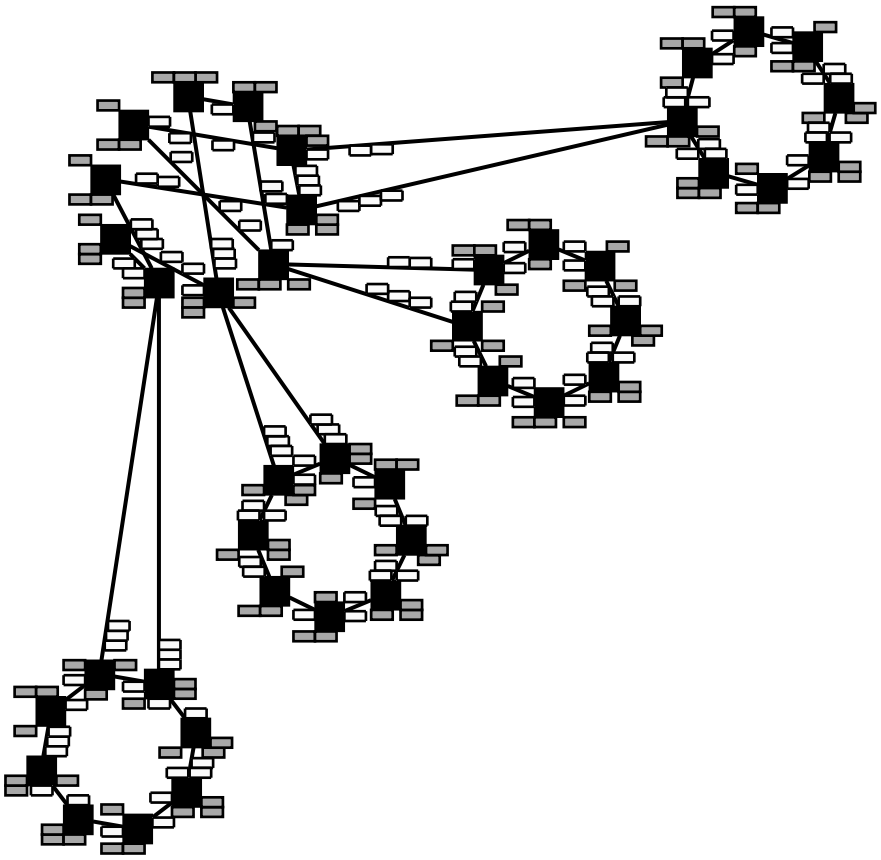

Figure 15.22 A circular drawing with three labels for each edge and node positioned by the Flow-based algorithm. The white boxes are edge labels and the dark boxes are node labels. Figure taken from [KT06].

15.3.5 Placing Labels by Modifying the Drawing

Automatic labeling is a very difficult problem, and because we rely on heuristics to solve it, there are cases where the best methods available do not always produce an acceptable or legible solution even if one exists. Furthermore, there are cases where no feasible solution exists. Given a specific drawing and labels of fixed size, then it might be impossible to assign labels without violating any of the basic rules of a good label assignment (e.g., label to label overlap, legibility, unambiguous assignment). These cases appear often in practical applications when drawings are dense, labels are oversized, or the label assignment must meet minimum requirements set by the user (e.g., font size or preference of placing labels).

To solve the labeling problem where the best solution we can have is either incomplete or not acceptable one must modify the drawing. This approach cannot be applied in drawings that represent geographical or technical maps where the underlying geometry is fixed by definition. However, the layout of a given graph drawing can be changed since it is the result of the algorithm used to draw the graph.

Generally speaking, there can be two algorithmic approaches in modifying the layout of a graph drawing:

- Modify the existing layout of a graph drawing to make room for the placement of labels.
- Produce a new layout of a graph drawing that integrates the layout and labeling process.

In [Hu09, KT11] algorithms that modify an existing layout of a graph drawing to make room for the placement of labels are presented. The algorithm of [KT11] modifies an existing orthogonal drawing by inserting extra space in order to accommodate the placement of edge labels that are free of overlaps. First, an edge label assignment is computed, where overlaps are allowed, by using existing techniques. Then, the drawing is modified by applying a polynomial time algorithm based on minimum flow techniques to find the extra space needed to eliminate label overlaps, while preserving the orthogonal representation of the drawing. In [Hu09] label overlaps are resolved by applying an algorithm based on the techniques used to produce force-directed layout drawings. It iteratively moves the labels to remove overlaps, while keeping the relative positions between them as close to those in the original layout as possible, and edges as straight as possible.

In [BDLN05, DDPP99, KM99] algorithms that combine the layout and labeling process of orthogonal drawings of graphs are presented. In [KM99] the authors study the problem of computing a grid drawing of an orthogonal representation of a graph with labeled nodes and minimum total edge length. They show an integer linear programming (ILP) formulation of the problem and present a branch-and-cut based algorithm that combines compaction and labeling techniques. The work in [BDLN05] makes a further step in the direction defined in [KM99] by integrating the topology-shape-metrics approach with algorithms for edge labeling. In [DDPP99] an approach to combining the layout and labeling process of orthogonal drawings is presented. Labels are modeled as dummy nodes and the topology-shape-metrics approach is applied to compute an orthogonal drawing where the dummy nodes are constrained to have fixed size.

References

[AF84] J. Ahn and H. Freeman. A program for automatic name placement. *Cartographica*, 21(2 & 3):101–109, 1984.

[AH95] D. H. Alexander and C. S. Hantman. Automating Linear Text Placement Within Dense Feature Networks. In *Proc. Auto-Carto 12*, pages 311–320. ACSM/ASPRS, Bethesda, 1995.

[AMO93] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[BDLN05] C. Binucci, W. Didimo, G. Liotta, and M. Nonato. Orthogonal Drawings of Graphs with Vertex and Edge Labels. *CGTA*, 32(2):71–114, 2005.

[C+99] Bernard Chazelle et al. Application challenges to computational geometry: CG impact task force report. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 407–463. American Mathematical Society, Providence, 1999.

[CMS95] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for Point Feature Label Placement. *ACM Trans. on Graphics*, 14(3):203–232, 1995.

[DDPP99] G. Di Battista, W. Didimo, M. Patrignani, and M. Pizzonia. Orthogonal and Quasi-upward Drawings with Vertices of Prescribed Size. In J. Kratochvil, editor, *Graph Drawing (Proc. GD '99)*, volume 1731 of *Lecture Notes in Computer Science*, pages 297–310. Springer-Verlag, 1999.

[DETT99] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ, 1999.

[DF92] J. S. Doerschler and H. Freeman. A rule based system for dense map name placement. *Communications of ACM*, 35(1):68–79, 1992.

[DKMT07] U. Doğrusöz, K. G. Kakoulis, B. Madden, and I. G. Tollis. On Labeling in Graph Visualization. Special Issue on Graph Theory and Applications, Information Sciences Journal, vol. 177/12, pp. 2459-2472, 2007.

[DMM+97] S. Doddi, M. V. Marathe, A. Mirzaian, B. M. Moret, and B. Zhu. Map Labeling and Its Generalizations. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 148–157, 1997.

[ECMS97] S. Edmondson, J. Christensen, J. Marks, and S. M. Schieber. A General Cartographic Labeling Algorithm. *Cartographica*, 33(4):321–342, 1997.

[Ede83a] H. Edelsbrunner. A new approach to rectangle intersections, Part I. *Internat. J. Comput. Math.*, 13:209–219, 1983.

[Ede83b] H. Edelsbrunner. A new approach to rectangle intersections, Part II. *Internat. J. Comput. Math.*, 13:221–229, 1983.

[EG90] L. R. Ebinger and A. M. Goulete. Noninteractive automated names placement for the 1990 decennial census. *Cartography and Geographic Information Systems*, 17(1):69–78, 1990.

[FA87] H. Freeman and J. Ahn. On the problem of placing names in a geographical map. *Int. J. of Pattern Rec. and Artificial Intelligence*, 1(1):121–140, 1987.

[FMC96] H. Freeman, S. Marrinan, and H. Chitalia. Automated Labeling of Soil Survey Maps. In *Proc. 8th Canadian Conference on Computational Ge-*

ometry, volume 1 of *Proceedings of the ASPRS-ACSM Annual Convention*, pages 51–59, 1996.

[Fre88] H. Freeman. An Expert System for the Automatic Placement of Names on a Geographic Map. *Information Sciences*, 45:367–378, 1988.

[Fre91] H. Freeman. Computer name placement. In D. J. Maguire, M. F. Goodchild, and D. W. Rhind, editors, *Geographical Information Systems: Principles and Applications*, pages 445–456. Longman, London, 1991.

[FW91] M. Formann and F. Wagner. A packing problem with applications to lettering of maps. In *Proc. 7th Annu. ACM Sympos. Comput. Geom.*, pages 281–288, 1991.

[GJS96] P. Gupta, R. Janardan, and M. Smid. Efficient Algorithms for Counting and Reporting Pairwise Intersections between Convex Polygons. In *Proc. 8th Canadian Conference on Computational Geometry*, pages 8–13. Carleton University Press, 1996.

[GK95] A. V. Goldberg and R. Kennedy. An Efficient Cost Scaling Algorithm for the Assignment Problem. *Mathematical Programming*, 71:153–178, 1995.

[Hir82] S. A. Hirsch. An algorithm for automatic name placement around point data. *The American Cartographer*, 9(1):5–17, 1982.

[Hu09] Y. Hu. Visualizing Graphs with Node and Edge Labels. *CoRR*, abs/0911.0626, 2009.

[Imh75] E. Imhof. Positioning names on maps. *The American Cartographer*, 2(2):128–144, 1975.

[KI88] T. Kato and H. Imai. The NP-completeness of the character placement problem of 2 or 3 degrees of freedom. In *Record of Joint Conference of Electrical and Electronic Engineers in Kyushu*, pages 11–18, 1988. In Japanese.

[KM99] G. W. Klau and P. Mutzel. Combining Graph Labeling and Compaction. In J. Kratochvil, editor, *Graph Drawing (Proc. GD '99)*, volume 1731 of *Lecture Notes in Computer Science*, pages 27–37. Springer-Verlag, 1999.

[KR92] D. Knuth and A. Raghunathan. The problem of compatible representatives. *SIAM J. Disc. Math.*, 5:36–47, 1992.

[KT98] K. G. Kakoulis and I. G. Tollis. An Algorithm for Labeling Edges of Hierarchical Drawings. In G. Di Battista, editor, *Graph Drawing (Proc. GD '97)*, volume 1353 of *Lecture Notes in Computer Science*, pages 169–180. Springer-Verlag, 1998.

[KT01] K. G. Kakoulis and I. G. Tollis. On the Complexity of the Edge Label Placement Problem. *Computational Geometry*, 18(1):1–17, 2001.

[KT03] K. G. Kakoulis and I. G. Tollis. A Unified Approach to Automatic Label Placement. *International Journal of Computational Geometry and Applications*, 13(1):23–60, 2003.

[KT06] K. G. Kakoulis and I. G. Tollis. Algorithms for the Multiple Label Placement Problem. *Computational Geometry*, 35(3):143–161, 2006.

[KT11] K. G. Kakoulis and I. G. Tollis. Placing Edge Labels by Modifying an Orthogonal Graph Drawing. In U. Brandes and S. Cornelsen, editors, *Graph Drawing (Proc. GD 2010)*, volume 6502 of *Lecture Notes in Computer Science*, pages 395–396. Springer-Verlag, 2011.

[MS91] J. Marks and S. Shieber. The computational complexity of cartographic label placement. Technical Report 05-91, Harvard University, 1991.

[PF96] I. Pinto and H. Freeman. The Feedback Approach to Cartographic Area Text Placement. In P. Perner, P. Wang, and A. Rosenfeld, editors, *Advances in Structural and Syntactical Pattern Recognition*, volume 1121 of *Lecture Notes in Computer Science*, pages 341–350. Springer-Verlag, 1996.

[RMM+95] 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.

[Tar83] R. E. Tarjan. *Data Structures and Network Algorithms*, volume 44 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1983.

[vKSW99] M. van Kreveld, T. Strijk, and A. Wolff. Point Labeling with Sliding Labels. *CGTA*, 13:21–47, 1999.

[vR89] J. W. van Roessel. An algorithm for locating candidate labeling boxes within a polygon. *The American Cartographer*, 16(3):201–209, 1989.

[WKvK+00] A. Wolff, L. Knipping, M. van Kreveld, T. Strijk, and P. K. Agarwal. A Simple and Efficient Algorithm for High-Quality Line Labeling. In P. M. Atkinson and D. J. Martin, editors, *Innovations in GIS VII: GeoComputation*, chapter 11, pages 147–159. Taylor and Francis, 2000.

[WMP+05] P. C. Wong, P. Mackey, K. Perrine, J. Eagan, H. Foote, and J. Thomas. Dynamic Visualization of Graphs with Extended Labels. In *Proceedings of the 2005 IEEE Symposium on Information Visualization*, INFOVIS '05, pages 73–80, 2005.

[WW95] F. Wagner and A. Wolff. Map Labeling Heuristics: Provably Good and Practically Useful. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 109–118, 1995.

[Yoe72] P. Yoeli. The logic of automated map lettering. *The Cartographic Journal*, 9(2):99–108, 12 1972.

[Zor90] S. Zoraster. The solution of large 0-1 integer programming problems encountered in automated cartography. *Operation Research*, 38(5):752–759, September-October 1990.