Pré-Publicações do Departamento de Matemática
Universidade de Coimbra
Preprint Number 06–15

A PRACTICAL MAP LABELING ALGORITHM UTILIZING IMAGE PROCESSING AND FORCE-DIRECTED METHODS

GEORG STADLER, TIBOR STEINER AND JÜRGEN BEIGLBÖCK

Abstract: Automatic placement of text corresponding to graphical objects is an important issue in several applications such as Geographical Data Systems (GIS), Cartography, and Graph Drawing. While usually only a finite number of possible placements is available, in this paper we allow for an infinite number of placements and only require the label to be as close as possible to its corresponding feature. We focus on realistic data and present a hybrid algorithm for labeling both line and point features. In the method's first step that works on the discretized map image processing tools are used to obtain an initial placement of all labels in allowed (i.e., non overlapping) position. The second step works on the continuous map and uses a force-directed iterative algorithm to improve this initial placement. In a comprehensive study on realistic data sets we investigate the performance of our method.

Keywords: automatic label placement, GIS-data, computational geometry, image processing, force-directed methods.

1. Introduction

Annotating realistic maps with pieces of text is an important problem in information visualization (see also the ACM Computational Geometry Task Force Report (Chazelle et al. 1999, [1]). Among others, it occurs frequently in automated cartography, i.e., in automated drawing of clear maps from geographical information science (GIS) data or state diagrams in technical drawings. Manual placing of map labels in visualizing systems yields optically appealing label arrangements, but this task is very laborious and thus usually too time-consuming. Since the data that has to be visualized proliferates rapidly, efficient algorithms for automatic label placements become more and more important. In order to obtain an overview over the current state of research for automatic map labeling we refer to the excellent bibliography (Wolff 2005, [14]).

There are many variants of labeling problems such as fixed position models with or without scalable labels, slider models, label number or size maximization problems. Complexity analysis reveals that the most interesting variants

Received March 14, 2006.

of these problems are NP-hard (we refer to the overview and the references given in Neyer (2001, [9]), Klau and Mutzel (2003, [8]), and Wolff (1999, [13]). Thus, to deal with these problems in practice, we must rely on good heuristic methods. This is especially the case if one has to deal with realistic problems arising in applications, where both point and line features have to be labeled.

Due to the large number of different problem formulations, several algorithms for map labeling can be found in literature. Comprehensive surveys of algorithms for map labeling are, e.g., Christensen et al. (1995, [2]) and Wagner and Wolff (1997, [12]). The methods usually rely on techniques such as greedy and exhaustive search algorithms, methods based on physical models (such as simulated annealing) and methods from integer programming. We refer to the papers mentioned above and to the selected contributions by Christensen et al. (1995, [2]), Ebner, Klau and Weiskircher (2003, [4]), Hirsch (1982, [6]), Klau and Mutzel (2003, [8]), Wolff (1999, [13]) and Wolff et al. (1999, [15]).

Here, we aim at an automatic map labeling algorithm that is able to deal with complicated maps arising, e.g., in geographical information system (GIS) applications. While usually for each label there is only a finite (usually small) number of possible placements, we consider an *infinite* number of possible label placements. In the problems we are dealing with both point and polygonal line features must be labeled, taking into account the following well accepted basic rules for map labeling:

- **Unambiguity:** Each label can be easily identified with exactly one graphical feature of the layout. It should be intuitively apparent to the reader of the map which label belongs to which point feature.
- **Avoidance of Overlaps:** Labels should not overlap with other labels or other graphical features of the layout.

Unlike in synthetic maps, in the practical applications we consider in this paper several additional requirements and adaptations of these basic rules have to be taken into account:

- *All* labels have to be placed, and *no scaling* of labels is admissible.
- Possibly *several labels* can belong to the same graphical feature.
- Each feature has an *infinite set of possible label positions.*The labels shall be placed as close as possible to the point features in order to maximize legibility of the maps.

- "Point features" are usually *boxes;* only in special cases these boxes can degenerate and become non-expanded boxes.
- Both label and point boxes are of *different size.*
- Besides point and line features, *other features can occur* that shall not overlap with labels as well, i.e., the regions for possible label placements might be further restricted.
- *Preferred label positions* (e.g., left or right to the object) have to be taken into account.

In this paper we present a hybrid approach for automatic label placing that consists of two steps. First, we work on a discretization of the map to obtain a collision-free initial position for each label. Then, this initial position is iteratively improved using a continuous force-based method.

To obtain an initial configuration we utilize techniques from image processing (see, e.g., the textbook by Sonka et al. (1998, [10])). The starting position is generated on a discrete map generated from pixelizing the map as a binary image. To avoid overlaps, regions around line and point features are excluded before a label is placed. In order to find feasible regions for labels we use a dilation technique for both point and line labels. This first step of our method results in a feasible configuration, i.e., besides possible discretization errors labels do not overlap among each other nor with features of the map. However, some labels are possibly far from their corresponding features. To improve the label placement obtained in this first step we apply an iterative force-directed strategy as a second step.

Force-directed methods have a rather long history in the context of graph drawing. Starting form the pioneering paper by Eades (1984, [3]), force-directed methods have attracted a considerable amount of research, we refer, e.g., to the book by Tollis et al. (1999, [11]) and the references given therein. These methods rely on the use of virtual forces between features and labels that lead to a clear and well legible label placement. Moreover, an advantage of force-directed methods is that they allow to implement aesthetic criteria in the system of virtual forces, which often leads to a good and visually clear label distribution.

However, to obtain good results, such a force-directed approach must be combined with a discrete method, e.g., with a simulated annealing algorithm. Such a hybrid approach has been followed in Ebner, Klau and Weiskircher (2003, [4]), where the label number maximization problem is addressed combining force-directed methods and simulated annealing.

Here, a related but different approach is used. We start with a feasible (or almost feasible) initial configuration generated by our discrete method and use virtual force vectors to improve this initial configuration. During the iteration process only few labels may overlap temporarily.

This paper is organized as follows. In the next section we introduce some notations and give basic definitions. In Section 3, our approach for finding an initial placement for all labels is presented. This initial position can be iteratively improved with the techniques presented in Section 4. In Section 5, we present our computational results on real world examples. In the concluding section 6, we draw conclusions and give an outlook for possible extensions of our methods.

2. Preliminaries

Here, we intend to explain the problem under consideration in more detail and introduce some notations and definitions. For the sake of simplicity, in the following presentation of our method we first restrict ourselves to the case that only point features have to be labeled and that at most one label has to be placed for each point feature. In this simplified problem line features are only treated as obstacles that should not be overlapped. After explaining our approach for this simplified problem, we will comment on the generalization of the approach for the problem in its full complexity, i.e., to problems where line features have to be labeled as well (horizontally and in parallel to the lines) and that several labels correspond to one feature. In Section 5 that contains our test examples we deal with real world problem data and thus with the the problem in its full complexity.

We now state the simplified problem that will be used to present our ideas. We intend to place $n \geq 1$ labels $\Lambda = (\lambda_1, \dots, \lambda_n)$ having width w_i and height h_i. The point features are $\Pi = (\pi_1, \dots, \pi_n, \pi_{n+1}, \dots, \pi_{n+k})$, where the label corresponding to π_i is λ_i for $1 \leq i \leq n$. The point features $\pi_{n+1}, \dots, \pi_{n+k}$ are not labeled but have to be taken into account as obstacles. As mentioned above, point features are rectangles, whose center points are located at the coordinates $\boldsymbol{x} = (x_i, y_i)$, and whose length and width are a_i and b_i, respectively. Moreover, the lines $(\ell_1, \dots, \ell_m)$ have to be taken into account as obstacles.

Since both, labels and point features are rectangles, we define an appropriate distance function to measure the distance between two rectangles. This *polygonal distance function* will be used in both steps of our method. For

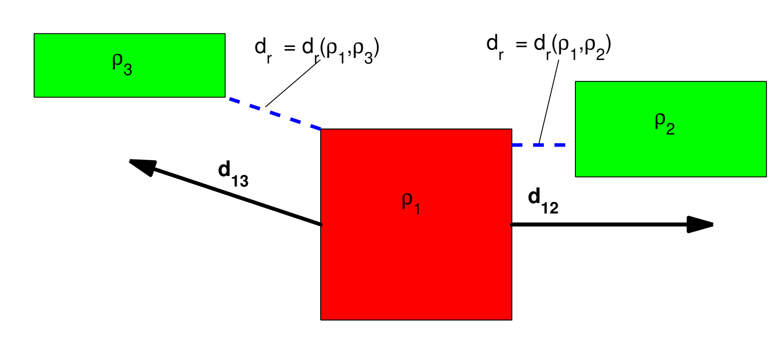

FIGURE 1. Visualization of the polygonal distance d_r and the unit length vector $\boldsymbol{d}$.

two rectangles ρ_1, ρ_2 we define the polygonal distance as

$$d_r = d_r(\rho_1, \rho_2) := \begin{cases} 0 & \text{if } \rho_1 \text{ and } \rho_2 \text{ overlap,} \\ \min\{\|p - d\| : p \in \rho_1, q \in \rho_2\} & \text{else.} \end{cases} \tag{1}$$

In the above definition, $\|\cdot\|$ denotes the Euclidean norm. The above definition is visualized in Figure 1, where also a vector $\boldsymbol{d}$ is drawn, that will be used in Section 4. The direction of this unit length vector is given by two points realizing the minimum in the above definition.

3. Initial positions

While calculating the initial positions of the feature labels, we heavily rely on image processing ideas. Since in applications many features and many regions that should be avoided have to be taken into account, finding admissible positions for each label is a highly nontrivial issue. As mentioned above, we first restrict ourselves to a simplified problem where only point features are labeled. Generalizations, for instance, the case that line features have to be labeled as well are discussed in Section 3.2

3.1. Label placement on a binary map. In order to find an initial configuration, we create a pixelized (binary) image $\Im$ of the reserved areas and the objects (points and polylines) for labeling. In $\Im$, discrete map positions marked with 0 ("white pixels") are free, i.e., not occupied by any object, while positions marked with 1 ("gray pixels") are occupied by an object or an already placed label. Whenever a label has been placed, $\Im$ is updated with its discrete position.

In order to create the pixmap $\Im$, we let the user choose a pixel size which can be selected according several aspects (e.g., map complexity, required label placement precision). For large maps of realistic complexity, $\Im$ must be split up into smaller rectangular regions. These image regions (tiles) need not to be held all in the core memory. They can be stored on a secondary medium, and only the currently necessary (few) tile(s) can be loaded into the core memory.

At hand of $\Im$, the following questions can be answered and operations be performed easily:

- Is a position (i, j) of $\Im$ free?
- Are all pixels belonging to a label free? (In this process, labels can be either parallel to the axis or can be in a general, rotated position.)
- Mark all pixels belonging to a polyline, box or label as free/occupied.
- Find one of the closest free positions to a specified image position (i, j). Here, one can either use discrete analogues of the usual Euclidean distance or of the polygonal distance as defined in (1). Restrict the search to positions that have a smaller distance from (i, j) than a user specified threshold.
- Perform morphological operations (dilation, erosion, etc.) on a rectangular cut-out of $\Im$. For the basics of mathematical morphology, its operations and applications we refer, e.g., to the textbook by Sonka et al. (1998, [10]).

Placing a point label λ as close as possible to a point feature π can be done as follows. We extract a square cut-out centered at π of the given map (see the left plot in Figure 2). Then, we create a binary pixmap of this cut-out (Figure 2, middle plot) and use the label we shall place as dilation structuring element (Figure 2, right plot). If now the label's midpoint is placed on one of the white (i.e., free) pixels, no overlaps with line or point features occur. We now choose one of the closest (either measured in Euclidean or the polygonal distance) free pixels to π and set all the pixels corresponding to the label λ to 1. Finally, the cut-out is copied back into $\Im$.

Normally, our method might detect several possible placements of comparable quality for each label. If a certain finite number of such placements is available, it might be useful to combine the search with a combinatorial technique such as simulated annealing (see, e.g., Kirkpatrick et al. (1983, [7])). However, this method can only deal with a finite number of possible positions

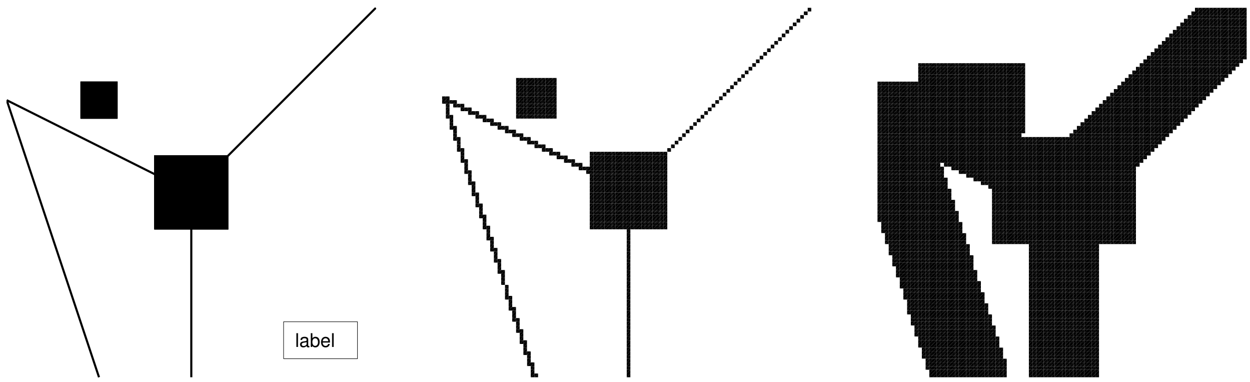

FIGURE 2. Procedure for initial placing of labels: Section of map and label to be placed (left), result after conversion into a binary map (middle) and result after dilation of the label (right)

for each label. For this reason, a step that detects possible placements (as described above) is always necessary.

3.2. Generalizations. At hand of the image $\Im$ and the algorithm for point feature labels, it is straightforward to find a collision free position for a label λ_l that is parallel to the coordinate axis and corresponds to a polyline ℓ.

Normally, there are several feasible positions for polyline labels. Then, we can select among those either using simulated annealing, or according to user preferences (e.g., labels shall be close to the middle of line segments, etc.).

In many maps, it is desirable that the line labels are placed parallel to one of the line segment. We assume at hand of the observation of practical data sets that line labels have a relatively small width compared to the lengths of the line segments or at least some of the line segments are comparably long as the label width. Therefore, we do not have to lean on more sophisticated approaches, e.g., splitting the label in at least two parts and placing every part parallel to different line segment (see for instance Wolff et al. (1999, [15])). Figure 3 shows the idea of placing a line label. In order to place a line label belonging to polyline, all polyline segments are potential candidates for bearing the line label. If the label collides an obstacle (e.g., a line obstacle in the figure), it is moved parallel to the polyline segment.

4. Iterative improvement using force-directed methods

To improve the label placement obtained with our discrete algorithm, we use an iterative force-directed method. That is, we define virtual forces

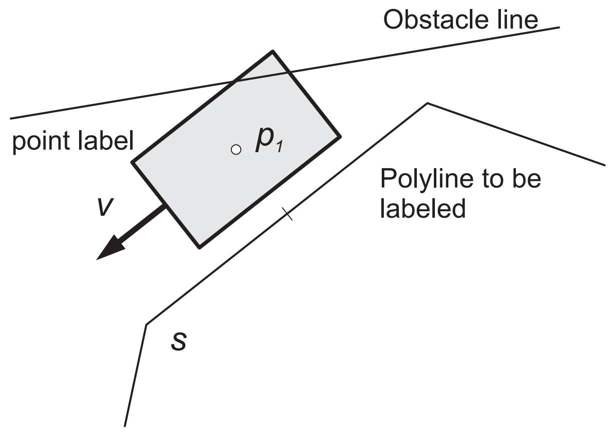

FIGURE 3. Due to an obstacle the line label must be shifted parallel to a line segment.

acting between labels, point and line features. These forces shall drag or push the labels into better positions. Already in 1982, Hirsch (1982, [6]) describes a strategy that uses virtual forces for moving a label on a circle around a point in order to obtain a collision-free placement. Enhancements of this approach are used in a patent held by Feigenbaum (1993, [5]), in which labels are placed as close as possible to corresponding point features by means of attractive as well repulsive forces acting between point features and labels. In this approach every label is initially of zero size and grows slowly to its original size. This shall enable movements in areas with a high density of labels. Our approach is related to this approach, but besides point features we have to label line features as well and, what makes the problem significantly more complex, lines shall not be overlapped by labels. Instead of shrinking the labels at the beginning of the iteration we rely on the initial configuration obtained by the methods described in the previous section. Another interesting contribution is the recent paper by Ebner, Klau and Weiskircher (2003, [4]), where a combination of a force-directed method and simulated annealing is used for a label number maximization problem.

4.1. Force-directed iterative algorithm. We now give a summary of our algorithm for improving the initial label placement. We assume that we are given a label placement that is feasible or almost feasible, i.e., only few overlaps occur (such a starting position can be obtained using the methods from the previous section) and we intend to improve this placement by means of a force-directed iterative algorithm. The method is sketched next.

Force-directed iterative algorithm (FDA)

(1) For each label λ_i ($1 \leq i \leq n$) centered at the coordinates $\boldsymbol{u}_i = (u_i, v_i)$ sum up the forces acting on this label. To be precise,
- Derive the virtual force vector $\boldsymbol{f}_{\lambda_i,\pi_i}$ that connects the label λ_i with the corresponding point feature π_i.
- Derive the repulsive force vectors $\boldsymbol{f}_{\lambda_i,\lambda_j}$ ($1 \leq j \leq n, j \neq i$) from the other labels, from point features $\boldsymbol{f}_{\lambda_i,\pi_j}$ ($1 \leq j \leq n+k$) and from line features $\boldsymbol{f}_{\lambda_i,\ell_l}$ ($1 \leq l \leq m$).
- Sum up all force vectors, i.e., derive

$$\boldsymbol{f}_i := \boldsymbol{f}_{\lambda_i,\pi_i} + \sum_{\substack{1 \leq j \leq n \\ j \neq i}} \boldsymbol{f}_{\lambda_i \lambda_j} + \sum_{1 \leq j \leq n+k} \boldsymbol{f}_{\lambda_i \pi_j} + \sum_{1 \leq l \leq m} \boldsymbol{f}_{\lambda_i \ell_l}.$$

(2) If all force vectors $\boldsymbol{f}_i, 1 \leq i \leq n$ are small enough or the maximum number of iterations is reached, stop the iteration. Otherwise perform a gradient-like step, i.e., update the label's centers $\boldsymbol{u}_i$ according to

$$\boldsymbol{u}_i := \boldsymbol{u}_i + \kappa \boldsymbol{f}_i$$

with an appropriate stepsize $\kappa > 0$.

(3) Update variables (such as for instance the step size) and go to Step 1.

The next subsection is mainly concerned with Step 1 of the above algorithm, i.e., with assembling the forces between different features. More details for the Steps 2 and 3 will be given in Section 4.4.

4.2. Description of the used forces. Here we explicitly describe the force vectors used in our implementation. Since we are dealing with rectangles as labels and point features, in the following we utilize the distance function d_r as defined in Section 2. We first start with defining the attractive forces between a label λ_i and its corresponding point feature π_i. We assume that the point and the label features are centered at $\boldsymbol{x} = (x_i, y_i)$ and $\boldsymbol{u} = (u_i, v_i)$, respectively. Then, we use the attractive force vector

$$\boldsymbol{f}_{\lambda_i,\pi_i} = \frac{d_r}{\|\boldsymbol{x} - \boldsymbol{u}\|}(\boldsymbol{x} - \boldsymbol{u}), \tag{2}$$

i.e., the force increases linearly as the label moves away from its corresponding feature. In the left plot in Figure 4 the above defined force vectors are visualized. On the right hand side of Figure 4 we show the repulsive force between a label and its corresponding point feature. Again we utilize the polygonal distance function d_r as defined in (1); As direction for the force we use the vector $\boldsymbol{d}$ defined by two points that realize the minimum in (1), see Section 2. If $d_r = 0$, that is, the two rectangles overlap, we choose for $\boldsymbol{d} := (\boldsymbol{u} - \boldsymbol{x})/\|\boldsymbol{u} - \boldsymbol{x}\|$. Moreover, we define $\tilde{d}_r := \max(\varepsilon, c_1 d_r)$ with $\varepsilon, c_1 > 0$. Then, the repulsing forces are given by

$$\boldsymbol{f}_{\lambda_i,\pi_i} := \left(\tilde{d}_r - 2 + 1/\tilde{d}_r\right)(\max(0, \operatorname{sgn}(1 - c_1 d_r)))\,\boldsymbol{d}. \tag{3}$$

Above, the parameter $0 < \varepsilon \ll 1$ has been introduced to obtain an upper bound for the repulsive force. This is needed to avoid possible difficulties in case of an overlap (i.e., in case that $d_r = 0$). For reasons of graphical representation this upper bound has been chosen relatively small for the right plot in Figure 4, in our implementation larger values are used. Moreover, in (3) the parameter c_1 allows one to control the size of the neighborhood where repulsive forces occur. Note that the middle term in (3) has the effect that repulsive forces have a compact area of support, i.e., they vanish outside a certain neighborhood of the point feature. To avoid zigzagging of (FDA) it is useful to have a smooth (at least C^1) transition along the boundary where the repulsive forces become zero. A brief calculation shows that the function given in (3) satisfies this requirement.

Observe in Figure 4 that the forces are well adopted to the rectangle shape of the features. To be precise, as direction for the repulsive forces we do not use the connection line between the rectangle's mid points but a direction that treats both space directions separately.

Clearly, analogous repulsive force as defined in (3) is used to avoid overlaps between labels with non-corresponding point features and labels among each other.

In many real world data sets one also has to take into account that labels shall not overlap with lines or, more generally, polygons. To deal with this problem we introduce repulsive forces between labels and polygons, see Figure 5. To derive the repulsive force for the labels we sum up the repulsive forces affecting all four corners of the label. To do so, we first calculate the footpoint (or the approximate footpoint) on the polygon for each corner. Then we evaluate a simple barrier-like 1D-function to obtain the repulsive force. As direction for the force we use the connection line between corner and corresponding footpoint on the polygon. Again, these 1D-repulsive forces have compact support and tend to zero smoothly.

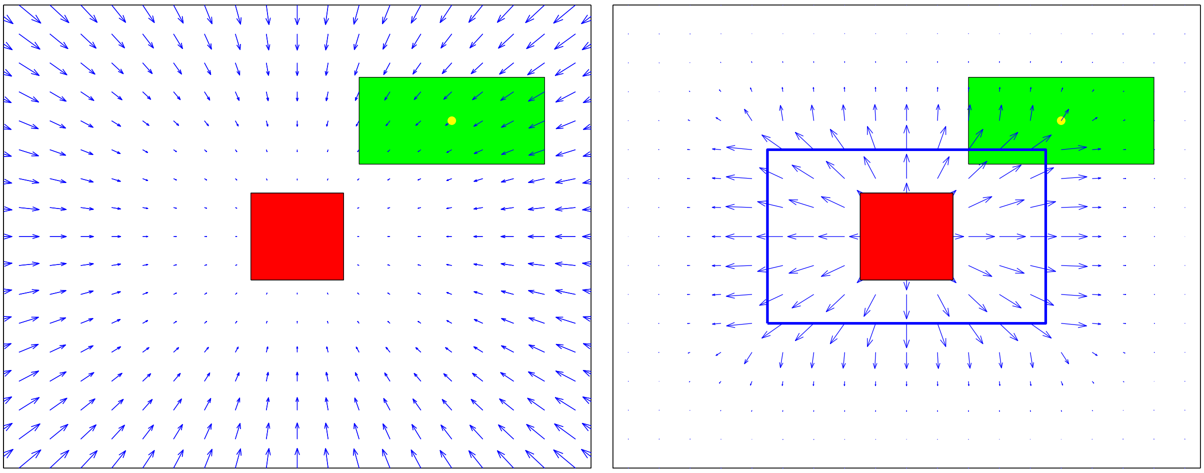

FIGURE 4. Forces (visualized by arrows) acting on the midpoint of label (light grey box) around point feature (dark grey box). Right: Attracting forces; Left: Repulsive forces; if the label's midpoint is inside the black box, features overlap.

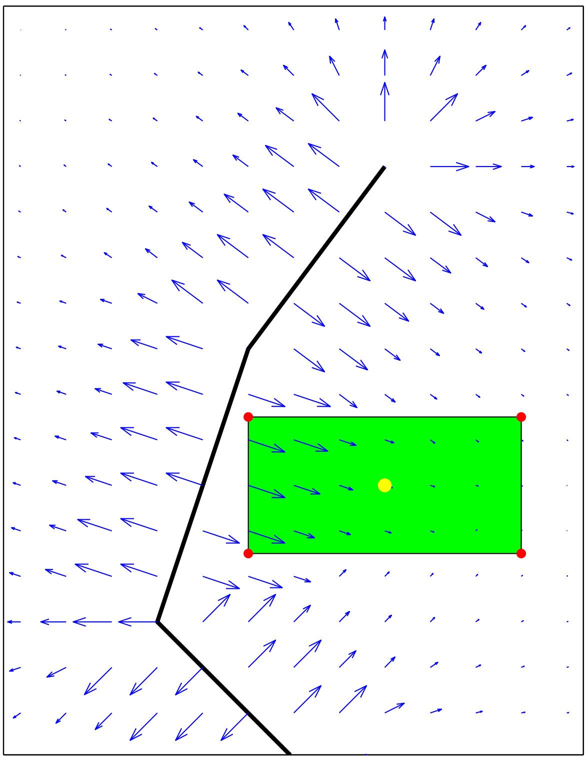

FIGURE 5. Forces (visualized by arrows) acting on the midpoint of label (light grey box) that repulse the label from a polygon (black line).

4.3. Generalizations in the presence of polygons. So far we have explained the forces needed to shift labels corresponding to point features. However, the generalization of the above ideas to (horizontal) labels corresponding to lines or to polygons is straight forward: Attractive forces between a label and the corresponding polygon are reduced to attractive forces between the label's midpoint and its footpoint on the polygon. The choice of repulsive forces to avoid overlaps is analogous as described above for labeling point features.

4.4. Some implementation details. In our implementation of the iterative methods described above we also use heuristics that have turned out to be useful in our numerical experiments.

Choice of the stepsize in (FDA). The choice of an appropriate steplength is a delicate issue. Too conservative stepsizes lead to a slow movement of the labels and therefore to a large number of needed iterations. Too large stepsizes may lead to overlaps between labels among each other or between labels and features. Though the algorithm is able to resolve such overlaps, they are clearly perturbing the convergence process since they cause the appearance of very large virtual forces. In our implementation we use the following simple strategy to adopt the stepsize: We start with a small value and slightly increase this value as long as no overlaps occur. If two features overlap, the stepsize is halved. Moreover, we use a lower and an upper bound for the stepsize.

Speeding up the force assembling. In each iteration of (FDA) one has to assemble all forces acting on a label. In principal, forces to all other labels and features have to be derived and summed up. However, since all repulsing forces have compact support and since we do not expect the configuration to change very quickly we can decrease the computational effort significantly: Only in the first iteration we test a larger number of force field and store those labels or features that have an influence (or are close to having an influence) on the label. In the following iterations only these (few) labels and features are taken into account. After a certain number of iterations one can again check a larger number of features and labels to update our list.

5. Computational Examples

Here we show some results obtained with our algorithms. We use GIS data and data taken from the visualization of telephone networks. To see the effects of our algorithms we show cutouts of maps before and after applying our methods.

5.1. Choice of parameters. In our implementation we use the following settings and parameters: The initial placement is usually done on a rather rough grid. The maximum number of iteration in the force-directed method is set to 30. Furthermore, we use $\varepsilon = 0.01$ and $c_1 = 3$.

5.2. Results. To begin with, we show in Figure 6 cutout of label placements for certain maps, where labels are indicated by rectangles. From Figure 6, one can already guess the abilities of our algorithms: The label placement resulting from our methods is quite good for maps with a moderate density of data (standard labeling situations). In these applications, there is usually enough free space for the force-directed method to move labels in a better position. For maps with an extremely large number of labels *and* many line features as obstacles, sometimes our force-directed method does not exhibit enough flexibility to rearrange the labels. For instance, in general a label cannot be tracked over a line and thus the initial placement already fixes the final position's range.

We now turn to a complex, realistic data set, in which 171 and 253 labels corresponding to point and label features, respectively, have to be placed. The data origins from a data base of a telephone company who is interested in a visualization of its networks. For the initial placement of the labels on the map whose approximate dimension is $1km$ times $1km$, a grid of 2000×2000 grid points has been chosen.

In Figure 7 we show a cutout of the initial configuration, where every label is on an initial (default) position. Here, the label's texts have been inserted in the corresponding rectangles and the map has been prepared using software for the visualization of GIS-data. Due to overlapping labels, the map is not legible, thus the resulting map is not suitable for delivering to a customer. (The visualization has been done with Autocad. For the sake of better legibility, the lines of the figure are thicker in the figure than they are in the original Autocad figure.)

Figure 8 shows the results of the application of our algorithm. Observe that the texts can be unambiguously assigned to the corresponding line and point features. In 8, all line labels are placed in horizontal position and not

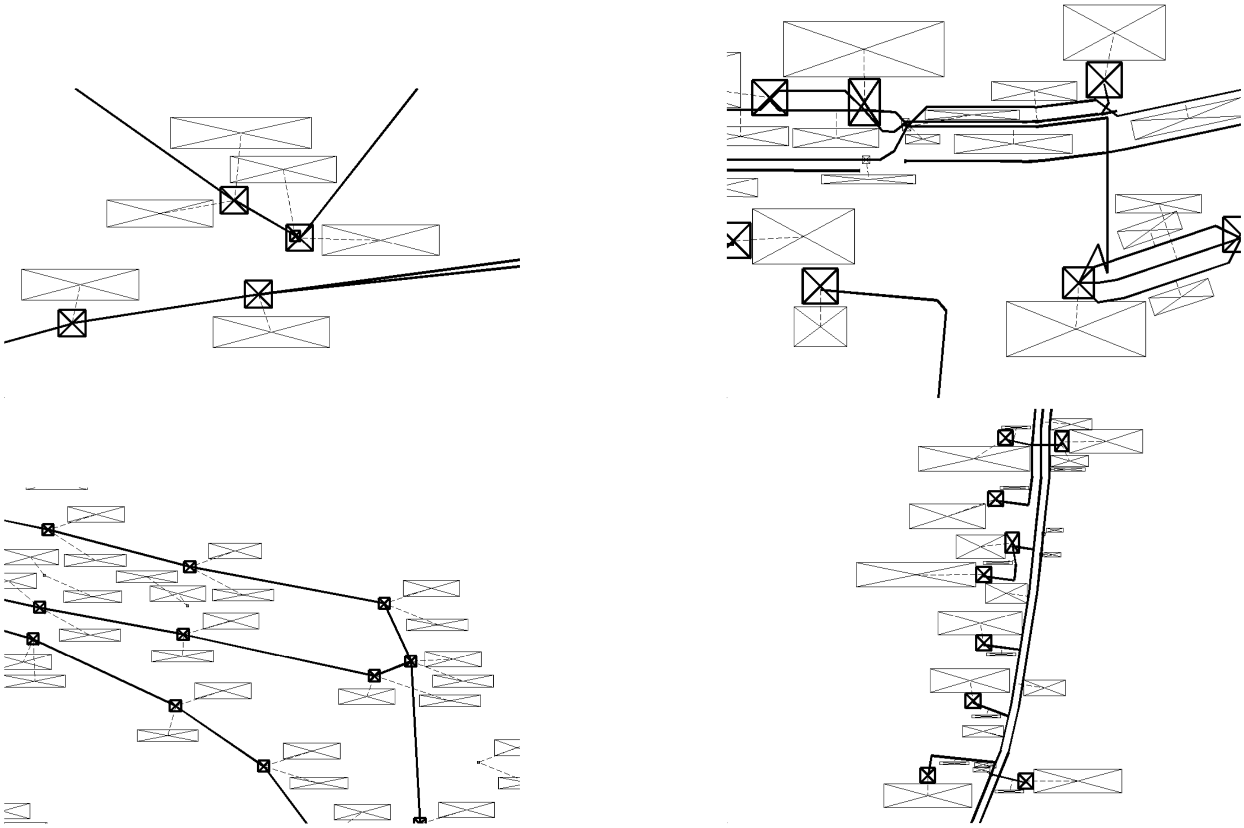

FIGURE 6. Cutouts of different maps; labels are represented by rectangles.

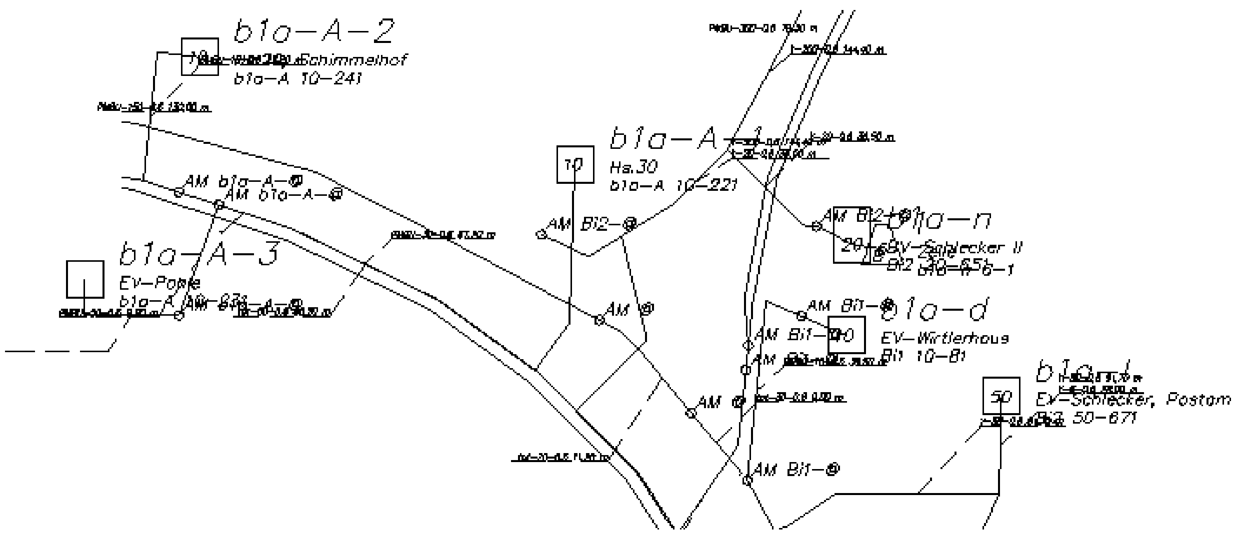

FIGURE 7. Cutout of map before label placement.

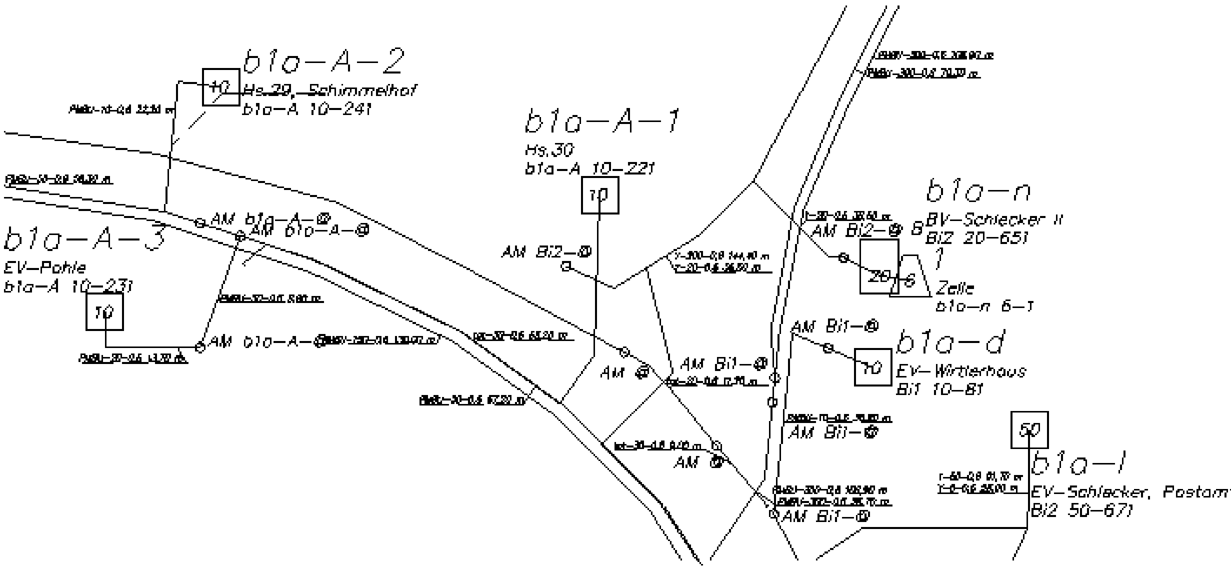

FIGURE 8. Same cutout as in Figure 7 after labels have been placed.

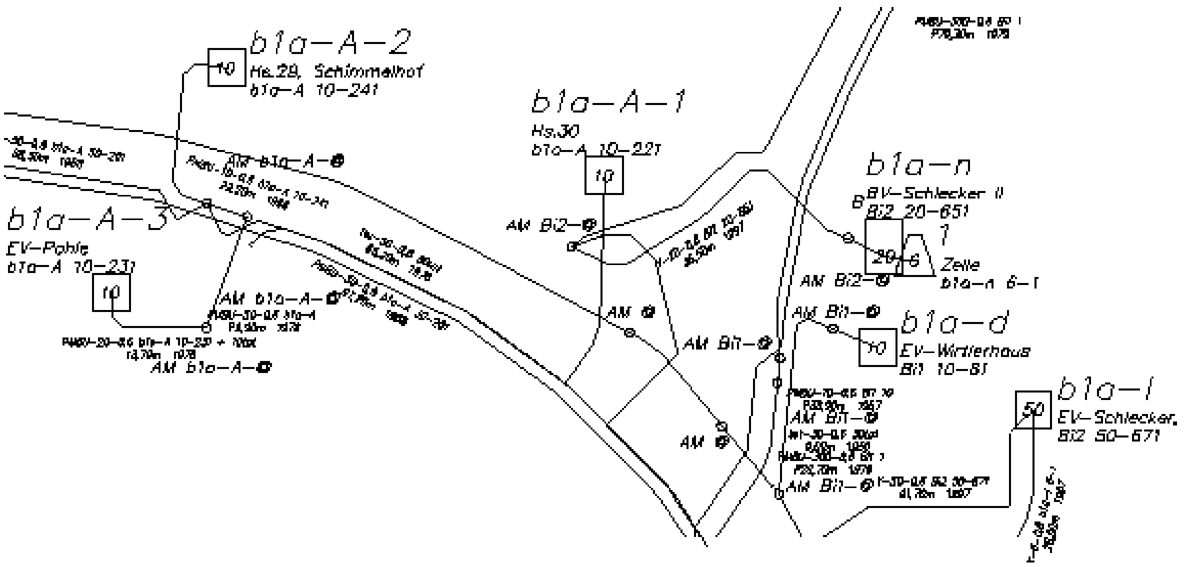

FIGURE 9. Same cutout as in Figure 8, but with line labels parallel to the corresponding line features.

parallel to one segment of the polyline. The labels are positioned collision free. The overall quality of placement suits well the practical requirements.

Finally, Figure 9 shows the same cutout as Figures 7 and 8, but now most line labels are parallel to the polylines. Comparing the result to 8 yields that this even improves the legibility of the map due to the fact that line labels can more easily be identified with the corresponding polylines.

6. Summary and Conclusions

The algorithm described in this paper has been developed to obtain reliable results in automatic map labeling. To the best of the authors' knowledge the combination between a discrete method based on image processing ideas and a continuous force-directed methods for automatic map labeling is a novel approach. Our test data contains, among others, GIS-data and data from telephone companies. In these applications many and possibly large regions of the map shall be avoided in order not to cover important information in the map. The results obtained with our hybrid method are of good quality, i.e., the labels do not overlap and can be uniquely assigned to the corresponding features. Thus, usually no manual adjustment of the labels is necessary anymore.

Acknowledgments

We are greatful to Helmut Pottmann (Vienna University of Technology) for helpful discussions and to Martin Peternell (Vienna University of Technology) and Martin Fink (rmDATA) for their kind assistance with the production of figures.

References

[1] B. Chazelle and 36 co-authors. The computational geometry impact task force report. In J. E. Goodman B. Chazelle and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223, pages 407–463. American Mathematical Society, Providence, 1999.

[2] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point-feature label placement. *ACM Trans. Graph.*, 14(3):203–232, 1995.

[3] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:146–160, 1984.

[4] D. Ebner, G. W. Klau, and R. Weiskircher. Force-based label number maximization. Technical Report TR–186–1–03–02, Vienna University of Technology, 2003.

[5] M. Feigenbaum. Method and apparatus for autmatically generating symbol images against a background image without collision utilizing distance-dependent attractive and repulsive forces in a computer simulation. United States Patent 5.355.314, 1993.

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

[7] S. Kirkpatrick and M. P. Vecchi C. D. Gelatt. Optimization by simulated annealing. *Science*, 220(4598):672–680, 1983.

[8] G. W. Klau and P. Mutzel. Optimal labelling of point features in rectangular labelling modells. *Mathematical Programming (Series B)*, 94:435–458, 2003.

[9] G. Neyer. Map labeling with application to graph drawing. In D. Wagner and M. Kaufmann, editors, *Drawing Graphs: Methods and Models*, volume 2025 of *Lecture Notes in Computer Science*, pages 247–273. Springer-Verlag, 2001.

[10] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision.* ITPS Thomson Learning, 1998.

[11] I. G. Tollis, G. Di Battista, P. Eades, and R. Tamassia. *Graph Drawing: Algorithms for the Visualization of Graphs*. Pretice Hall, 1999.

[12] F. Wagner and A. Wolff. A practical map labeling algorithm. *Computational Geometry: Theory and Applications*, 7:387–404, 1997.

[13] A. Wolff. *Automated Label Placement in Theory and Practice*. PhD thesis, FU Berlin, 1999.

[14] A. Wolff. The map labeling bibliography. URL location: `http://i11www.ilkd.uni-karlsruhe.de/~awolff/map-labeling/`, 2005.

[15] A. Wolff, L. Knipping, M. van Kreveld, T. Strijk, and P. K. Agarwal. A simple and efficient algorithm for high-quality line labeling. In D. Martin and Fulong Wu, editors, *Proc. GIS Research UK 7th Annual Conference (GISRUK'99)*, pages 146–150, 1999.

Georg Stadler
Center of Mathematics, University of Coimbra, Apartado 3008, 3001-454 Coimbra, Portugal

E-mail address: `georgst@mat.uc.pt`

Tibor Steiner
Geometric Modelling and Industrial Geometry, Vienna University of Technology, Wiedner Hauptstrasse 8–10, 1040 Vienna, Austria

E-mail address: `tibor@geometrie.tuwien.ac.at`

Jürgen Beiglböck
rmDATA Datenverarbeitungsges.m.b.H., Prinz-Eugen-Str. 12, 7400 Oberwart, Austria

E-mail address: `beiglboeck@rmdata.at`