TECHNISCHE UNIVERSITÄT WIEN
Institut für Computergraphik und Algorithmen

Force-Based Label Number Maximization

Dietmar Ebner, Gunnar W. Klau and René Weiskircher

Forschungsbericht / Technical Report

TR–186–1–03–02

5. Juni 2003

Favoritenstraße 9-11 / E186, A-1040 Wien, Austria
Tel. +43 (1) 58801-18601, Fax +43 (1) 58801-18699

Force-Based Label Number Maximization

Dietmar Ebner, Gunnar W. Klau, and René Weiskircher

Institute of Computer Graphics and Algorithms, Vienna University of Technology,
http://www.ads.tuwien.ac.at, ebner|gunnar|weiskircher@ads.tuwien.ac.at

Abstract. We present a force-based simulated annealing algorithm to heuristically solve the *NP*-hard label number maximization problem LNM: Given a set of rectangular labels, each of which belongs to a point-feature in the plane, the task is to find a *labeling* for a largest subset of the labels. A labeling is a placement such that none of the labels overlap and each is placed at its point-feature.
The abstraction of the problem to the virtual force system allows us to implement additional aesthetic criteria and to compute placements with good label distribution in a short amount of time. Furthermore, our algorithm can be applied as a postprocessing step to improve existing labelings. We find that the results often look similar to those of a human cartographer.
Extensive computational experiments on widely used benchmark data demonstrate that our new algorithm produces solutions where the number of placed labels is close to optimality and where the distribution of the labels is better than in labelings computed by algorithms that only maximize the number of placed labels. The experiments also show that the algorithm is able to solve large problems fast. This demonstrates its applicability in dynamic real-time environments, *e.g.*, in navigation systems.

1 Introduction

Automated map labeling attracts many researchers in computer science due to its numerous applications, *e.g.*, in cartography, geographic information systems, point pattern analysis, spatial statistics, and graphical interfaces.

The growing amount of data for which informational graphics have to be produced leads to an increasing need for automatic labeling procedures. Due to the complexity of the underlying problems, manual map labeling is a tedious and time-consuming task. In addition to the classical problem of labeling a two-dimensional cartographic map, labeling problems arise in geographic information systems, navigation systems, and fully automatically generated technical maps where manual labeling is impossible.

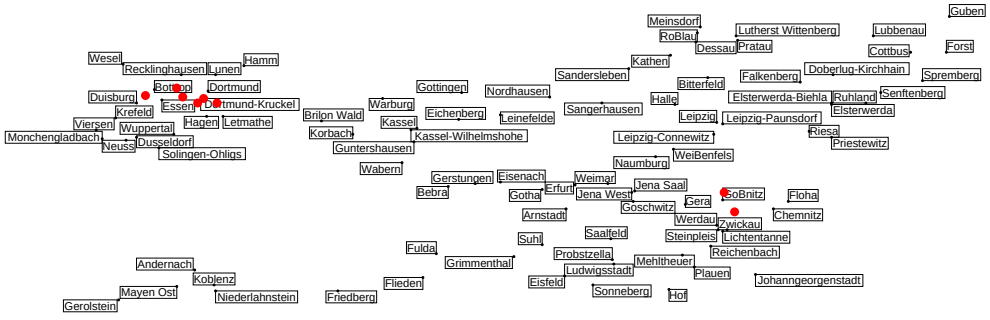


Fig. 1. Sample labeling produced with our algorithm in the slider model. Thicker points represent cities whose labels could not be placed.

1.1 Point-Feature Map Labeling

A major problem in map labeling is the *point-feature label placement* where the task is to place labels adjacent to point-features so that only a few or no labels overlap. These features may be cities, mountain peaks, or points in a plot that represent statistical data. Besides point-features, also area-features, like countries and seas, or line-features, like streets and rivers, can be labeled. In this paper we focus on point-feature labeling and restrict the labels to be iso-oriented axis-parallel rectangles; the bibliography [13] maintained by Wolff is a good starting point for literature on other models.

Several criteria have been developed that characterize a high-quality labeling:

(C1) On a good map the placement of labels is as unambiguous as possible. It is intuitively apparent to the reader of the map which label belongs to which point-feature. This implies that labels are close to the point-features they belong to.
(C2) The information of the labels is legible. Unambiguity alone does not help if the user cannot read the text in the labels.
(C3) No or only a few labels overlap. Obviously, overlaps decrease the legibility of a map.
(C4) The number of omitted labels is low.

The cartographic literature contains more rules, see, *e.g.*, the papers by Imhof [5] and Yoeli [14]. Yet, the overall aim in automatic map labeling is to devise algorithms that produce labelings of maximum legibility.

An instance of a labeling problem consists of a set of point-features, information about the label sizes, and a mapping from labels to point-features. In general it is not possible to place all the given labels in their original size without any overlap. The literature suggests several possibilities to deal with this problem; among these are decreasing the size of the labels to allow a placement of all labels without any overlap, and keeping the sizes of the labels fix while looking for the maximum number of labels that can be placed. The first possibility is referred to as the *label size maximization problem*, the second one as the *label number maximization problem*.

Research in automated map labeling has mainly focused on the six *labeling models* shown in Figure 2, the most popular of which are the four-position and the four-slider model. The dots in the figure represent the point-feature to be labeled.

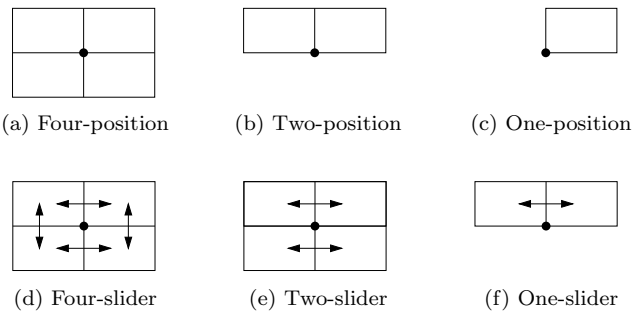


Fig. 2. Axis-parallel rectangular labeling models. A label can be placed in any of the positions indicated by the rectangles and can slide in the directions of the arrows

Definition 1 (Label Number Maximization problem). *Given a set* $\Lambda = \{\lambda_1, \ldots, \lambda_k\}$ *(the labels), two functions* $w, h : L \to \mathbb{R}$ *(the widths and heights of the labels), and a function* $a : \Lambda \to \mathbb{R}^2$ *(the points to be labeled), find a subset* $\Lambda_P \subseteq \Lambda$ *of largest cardinality and a*

function $\rho : \Lambda_P \to \mathbf{R}$, where $\mathbf{R}$ is the set of axis-parallel rectangles in the plane, so that the following conditions hold:

(L1) Rectangle $\rho(\lambda)$ has width $w(\lambda)$ and height $h(\lambda)$ for every $\lambda \in \Lambda_P$.
(L2) Point $a(\lambda)$ lies on the boundary of $\rho(\lambda)$ for all $\lambda \in \Lambda_P$
(L3) The open intersection $\rho(\lambda) \cap \rho(\mu)$ is empty for all $\lambda, \mu \in \Lambda_P$, $\lambda \neq \mu$.

An assignment of labels to rectangles that satisfies the three properties (L1)–(L3) is called a *labeling*. Properties (L1) and (L2) make sure that each label λ is drawn with the given size and attached correctly to its point-feature $a(\lambda)$. Property (L3) forbids overlaps between the labels.

1.2 Previous Work

Force-based methods in graph drawing. Force-directed methods have originally been developed for drawing graphs in a structured and human readable way. In practice, these techniques often perform remarkably well on medium-sized instances and are easy to implement. Further, the resulting drawings typically capture symmetries while avoiding the expensive computations to look for them explicitly.

These algorithms, going back to Eades [3] and Kruskal and Seery [9], view the input graph as a system of objects with forces acting between them. Configurations of the objects with low energy correspond to aesthetically pleasing layouts of the graph. Algorithms for this task are mostly variations of iterative gradient-based methods such as the Newton-Raphson method.

Davidson and Harel consider in [2] the number of edge crossings in a drawing as an additional, discrete term in the objective function and can therefore not apply gradient methods to find an equilibrium. The authors propose the *simulated annealing* approach introduced in [6] by Kirkpatrick, Gelatt Jr., and Vecchi. This approach defines for each configuration a finite set of neighboring configurations and tries one of them at random. New configurations are always accepted if they decrease the energy of the system but even if they increase the energy, they are accepted with a probability that decreases with time. As we will point out in Section 2, we will use simulated annealing for similar reasons as Davidson and Harel.

Point-feature map labeling. Most previous work on automated point-feature map labeling concentrates on the discrete models, where a label can only allocate a finite number of positions with respect to its point-feature. See the bibliography [13] for an overview.

Van Kreveld, Strijk, and Wolff [12] define several variations of slider models, namely, the one-, two-, and four-slider model (see Figure 2) and show *NP*-completeness of the decision problem in the four-slider model (independently, Marks and Shieber have shown this in [10]). The main result in [12] is a $\frac{1}{2}$-approximation algorithm that is able to find a solution of LNM in any of the slider-models with unit height rectangles. The algorithm is a $\Theta(n \log n)$ greedy sweep-line algorithm. For the same models, the authors develop a polynomial time approximation scheme. The respective algorithms label at least $(1 - \varepsilon)$ times the optimum number in overall running time $O(n^{4/\varepsilon^2})$. Strijk and van Kreveld extend the above mentioned $\frac{1}{2}$-approximation algorithm for the slider models in [11] to labels with different heights. If r denotes the number of different label heights, the running time of the algorithm is $O(rn \log n)$.

Klau and Mutzel present in [8] an exact algorithm for the label number maximization problem that works in any of the labeling models. The method is based on a pair of so-called constraint graphs that code horizontal and vertical positioning relations. The key idea is to link the two graphs by a set of additional constraints, thus characterizing all feasible solutions

of LNM. This enables the formulation of a zero-one integer linear program whose solution leads to an optimal labeling.

The paper [1] by Christensen, Marks, and Shieber contains an extensive computational study of labeling methods in the four-position model. The authors also present a simulated annealing method for this problem that is the clear winner of the study in terms of labeling quality while still being resonably fast. Furthermore, they propose a procedure for randomly creating labeling instances. We use this benchmark generator, which has become a widely used tool in map labeling research, for our computational experiments in Section 3.

Force-based Methods in Map Labeling. Already in 1982, Hirsch introduced a model that is similar to the four-slider model and proposed an algorithmic labeling method that can be interpreted as a force-directed approach. The algorithm starts with an inital label placement and tests for overlaps. Based on the amount of intersecting area, overlap vectors are computed for labels involved in an overlap conflict. For each label, the summation of these vectors helps in heuristically deciding where to move the label. Successive movements of all labels in conflict, which Hirsch calls a *map sweep*, is done by using one of the following two methods: (a) Moving labels in the direction of the computed vectors with sequential stops at preferred positions. This method allows the label to be placed at any possible position. (b) Performing a discrete jump to a position indicated by the vector angle. Here, the primary aim is to solve an overlap situation where the first method fails.

Hirsch does not consider the number maximization problem explicitly, but rather proposes interactive conflict resolution with the help of an experienced cartographer. Also, although his overlap vectors resemble the intersection-proportional component within our force system, he does not consider distance-related forces and suggests a different method for finding an equilibrium of minimum energy. His approach can be seen as a gradient-driven heuristic.

In a way, Hirsch's ideas are also applied in [4], a patent specification authored by Feigenbaum. The problem considered in the patent is related to, but significantly different from, the label number maximization problem: Here, labels should be placed as close as possible to the corresponding point-features. The author defines forces that attract each label to its associated point-feature and repulsive forces that reject labels from other features on the map. In contrast to our labeling model, labels in the referred patent should be placed as near as possible to their features, which allows a significantly higher number of labels to be placed. The described system searches its final state by applying incremental moves according to the calculated forces. Every label has initially zero size and grows slowly to its original dimensions to facilitate movements in dense areas.

1.3 Contribution and Overview

We present a new and very efficient heuristical approach in the most general of the labeling models, the four-slider model. Our new method is hybrid in the sense that it is based both on an underlying system of forces and a higher-ranking simulated annealing method. The abstraction of the problem to the virtual force system allows us to implement additional aesthetic criteria and to compute placements with good label distribution in a short amount of time. Furthermore, our algorithm can be applied as a postprocessing step to improve existing labelings. We find that the results often look similar to those of a human cartographer. Figure 1 on the first page of this paper shows a typical output as produced by our algorithm.

The rest of this paper is organized as follows: Section 2 presents the new approach. We start by explaining the underlying force model and point out that it is not sufficient to use a classical approach to find a state of low energy within this system. We propose to combine the force directed method with simulated annealing to overcome this drawback and conclude the

section by presenting our hybrid approach. Our experimental results in Section 3 show that our algorithm produces very good results very efficiently. To test the quality of the solutions computed by our new method, we have compared its results to optimal solutions of a large set of benchmark labeling instances. We also show that the distribution of the labels is much better than in solutions computed by an algorithm that only optimizes the number of labels placed. Finally, Section 4 concludes the paper and presents possible extensions and future lines of research.

2 Force-Directed Map Labeling

In this section we describe our force-based simulated annealing algorithm for the label number maximization problem. Our approach uses repulsive forces between labels, which are used to compute a force vector for each label. The length and direction of these vectors gives us an idea where to place individual labels and how to solve potential conflicts between two or more labels.

As a side-effect we achieve another important benefit, which makes the method usable for practical applications: Our forces are defined to grow super linearly with decreasing distance between two labels. Therefore, labels are not placed close to each other if possible and the method achieves a good distribution of the labels in the available space. This improves the readability of the labels and results in an aesthetically pleasing arrangement. We find that our results often look similar to those of a human cartographer.

To avoid being trapped in local minima of the energy function, we combine the purely force directed method with the simulated annealing approach. A short overview of this method is given in Section 2.3.

2.1 Designing the Force Model

Every force-directed algorithm consists of two major parts: (a) a force-system between the objects and (b) a method that seeks an equilibrium of minimum energy.

In our case a low energy equilibrium configuration should correspond to a pleasing labeling. In contrast to applications in graph drawing our labels are bound to their point-feature and may not be positioned freely in the available space. Each label must be placed in such a way that it touches its point-feature and does not overlap other labels. Since we only allow (intermediate) positions that satisfy at least the first condition, we do not need any attractive forces between a point and its associated rectangle. We restrict the computation of forces to pairs of labels that might intersect. We call the set of those labels for each label λ the *neighborhood* $N(\lambda)$.

Definition 2. *The* neighborhood *of a label $\lambda \in \Lambda$ with width $w(\lambda)$, height $h(\lambda)$ and associated point $a(\lambda) = (x_\lambda, y_\lambda)$ is*

$$N(\lambda) = \{\mu \in \Lambda \mid w(\lambda) + w(\mu) \geq |x_\lambda - x_\mu| \wedge h(\lambda) + h(\mu) \geq |y_\lambda - y_\mu|\} \ .$$

Our main goal is to place as many labels as possible in the available space without any intersections. Therefore the decisive factor in our force system is the amount of intersection between two labels. We call this force the *intersection-proportional* component. The amount of the second force acting in our model, the so called *distance-related* part, depends on the distance between two labels and grows, if two rectangles are placed close to each other. The distance-related part is not the significant value in our model, its only purpose is to guide the algorithm to a well distributed labeling.

Definition 3. *For every two labels $\lambda, \mu \in \Lambda$, we define $d_{\min} : \Lambda \times \Lambda \to \mathbb{R}$ as*

$$d_{\min}(\lambda, \mu) = \begin{cases} 0 & \text{if } \lambda \text{ and } \mu \text{ overlap} \\ \min\{\|p, q\| \mid p \in \lambda, q \in \mu\} & \text{otherwise .} \end{cases}$$

The function $\|p, q\| : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ denotes the Euclidean distance between the two points p and q in the Euclidean plane $\mathbb{R}^2$.

Further we define $i_x(\lambda, \mu), i_y(\lambda, \mu) : \Lambda \times \Lambda \to \mathbb{R}$ as the amount of intersection between labels λ and μ in horizontal, respective vertical, direction.

We can now define the force function $f = (f_x, f_y)$ for each label in the following way:

Definition 4. *For each label $l \in \Lambda$ with center point $c_l = (x_l, y_l)$, the x-component of the force function $f_x : \Lambda \to \mathbb{R}$ is defined as*

$$f_x(l) = \sum_{r \in N(l)} \frac{(f_i(l, r) + f_d(l, r))\ (x_l - x_r)}{\|c_l, c_r\|} \ . \tag{1}$$

The functions $f_i : \Lambda \times \Lambda \to \mathbb{R}$ and $f_d : \Lambda \times \Lambda \to \mathbb{R}$ are defined as follows:

$$f_i(l, r) = \delta_1\, i_x(l, r)\, i_y(l, r) \tag{2}$$

$$f_d(l, r) = \frac{\delta_2}{\max(\varepsilon, d_{\min}(l, r))^2} \ . \tag{3}$$

The constants $\delta_1, \delta_2 \in \mathbb{R}$ control the influence of the particular term on the force function f_x. In the following f_i will be called the intersection-proportional *component and f_d will be referred to as the* distance-related *part of the force. We define the* y-component f_y *analogously.*

Note that the direction of the force between two labels is defined by the location of their center points and that ε limits the amount of f_d to a value of δ_2/ε^2.

2.2 Drawbacks of a Purely Force Directed Method

A purely force directed method performs poorly if the labels take a significant fraction of the available drawing area. There is only little space for manoeuvre when seeking an equilibrium, especially if incremental methods are used. In the area of map labeling, most practical examples consist of dense drawings and do not leave much space for moving labels around without producing new intersections. The problem is aggravated by the fact that we only allow horizontal and vertical moves around the label's border.

Figure 3 shows an example of a bad local minimum that is difficult to escape from by using incremental moves. It is not possible to transform the bad labeling on the left continously into the good labeling on the right without a temporary increase of overlaps and thus of the overall energy of the system.

Another problem arises from the direction of our forces. We designed them in Section 2.1 to act between the center points of two labels. But since labels have non uniform size and they are bound to their point-features, the direction of our resulting force vector does not always indicate a solution for the conflict. Figure 4 shows a very simple example consisting of just two point-features. Any algorithm that strictly follows the direction of the force vector is not able to resolve the shown configuration.

Therefore, we need to accept worse intermediate configurations to be able to escape local minima and we propose to use the simulated annealing method for this purpose. Details of the adaptation of the general simulated annealing algorithm to our needs are described in the next section.

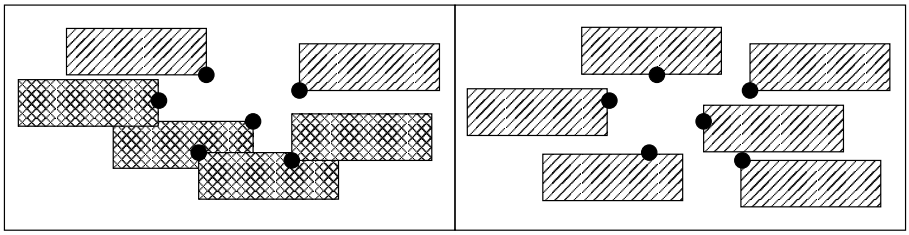


Fig. 3. A bad local minimum and an optimal labeling of the same instance

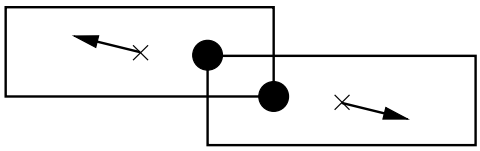


Fig. 4. Example configuration where the forces cannot resolve an overlap

2.3 A Short Summary of Simulated Annealing

Simulated annealing is a very flexible optimization method, which can be used in a wide range of combinatorial optimization problems. It is derived from the following observation: When cooling down a liquid rapidly to a crystal form, the system results in amorphous structures with a high energy while slow cooling results in a crystal structure with lower energy.

The general simulated annealing procedure applies a series of sequential moves while simultaneously decreasing the temperature. The main idea is that the probability with which the change from a state with energy E_1 to a state with energy E_2 will be accepted is $e^{-\frac{E_2-E_1}{k\,T}}$, where k is a positive constant. Thus the probability for moves that increase the energy decreases with a falling temperature. Algorithm 1 shows an overview of the simulated annealing method.

2.4 The Hybrid Algorithm

We now present the hybrid force-based simulated annealing algorithm for the label number maximization problem and describe some of its implementational details. Note that many parts of the algorithm can be easily replaced by similar methods. This makes it easy to adopt the algorithm to perform better on some special kinds of input maps. Furthermore

Algorithm 1 Overview of the simulated annealing method

```
1: compute initial configuration σ and initial temperature T;
2: repeat
3:    choose new configuration σ̂ from the neighborhood of σ;
4:    E ← c(σ); Ê ← c(σ̂);                    {c is a cost function that evaluates configurations}
5:    r ← random value ∈ [0...1];
6:    if σ̂ < σ ∨ r < e^((E−Ê)/T) then
7:       σ ← σ̂;
8:    decrease temperature T;
9: until termination rule is satisfied
```

the parameters like the constants δ_1, δ_2 in our force model, the initial temperature T_{init} or the specific cooling scheme have an important impact on the quality of the produced solutions.

The main loop of the algorithm is executed until all overlaps are resolved. Inside this loop, the simulated annealing method is executed with a loop that runs through all labels that have not yet been thrown out as unplaceable. In contrast to a standard simulated annealing implementation, the algorithm first uses the forces to move the labels towards a configuration with low energy. If there are still intersections, labels involved in the intersection are moved to a randomly chosen position out of a small set of standard positions. The new position is always accepted if it decreases the energy and may be accepted if it does not increase the energy by more than the current temperature allows.

After each label has been checked for a better position, the temperature is decreased and the next iteration of the loop begins. If the temperature reaches a lower threshold or there are no more overlaps, the inner loop ends. If overlaps are still present, a heuristically chosen label involved in overlaps is removed. Now the temperature is reset to a predefined reset value and the simulated annealing method starts again. Algorithm 2 shows the algorithm in pseudocode.

Algorithm 2 Force-directed map labeling

$T \leftarrow T_{\text{init}}$;
$\sigma \leftarrow$ initial configuration;
repeat
  $T \leftarrow \max(T, T_{\text{reset}})$
  repeat
    for all remaining labels $\lambda \in \sigma$ **do**
      $\hat{\sigma} \leftarrow \sigma$;
      if $|f_x(\lambda)| > F_{\min} \vee |f_y(\lambda)| > F_{\min}$ **then**
        {*try to reach an equilibrium in horizontal and vertical direction*}
        move λ according to its force vector until an equilibrium is reached or no further moves are possible;
        update force vector on all labels $\in N(\lambda)$;
      if $\exists \mu \in N(\lambda)$ such that $i_x(\lambda, \mu) > 0 \wedge i_y(\lambda, \mu) > 0$ **then**
        {*the intersection could not be resolved*}
        move λ to a completely different position independent of its force vector;
        update force vector on all labels $\in N(\lambda)$;
    if $\sigma > \hat{\sigma} \wedge$ random value $r \in [0 \dots 1] \geq e^{\frac{\sigma - \hat{\sigma}}{T}}$ **then**
      $\sigma \leftarrow \hat{\sigma}$; {*reject modified configuration*}
    decrease temperature T according to the cooling scheme;
  until $T \simeq 0 \vee$ no more labels overlap;
  if at least two labels still intersect each other **then**
    select label μ according to a suitable heuristic;
    $\sigma \leftarrow \sigma \setminus \{\mu\}$;
    update force vector on all labels in the neighborhood;
until σ does not contain intersecting labels;

The following remarks explain some parts of the algorithm in more detail:

– Most force-directed applications in graph drawing use numerical algorithms like the Newton-Raphson method or similar techniques when searching for an equilibrium for each node. These methods can't be used in our algorithm due to the following reasons:
  • In contrast to zero sized nodes, labels must be placed according their associated point-feature. A position that corresponds to an equilibrium of the forces is not always valid in respect to the point.

- Most methods require at least the first derivation of the function. Our forces depend on a combination of the overlapping area and the distance between two labels, which are both defined differently depending on the specific domain, and are thus not continuous.

  A simple algorithm could move the label either a constant amount or proportional to the force vector in the particular direction. To speed up convergence we implemented a binary search variant that finds an equilibrium in pseudo-polynomial time.
- When searching for an alternative position in line 14 of the algorithm, we realized that moving the label randomly to a position out of a small set of predefined positions is sufficient. We compared this simple and very fast method with more sophisticated approaches but computational experiments showed that the increase in run time caused by more elaborate methods does not produce a corresponding increase in the quality of the solutions.
- We have chosen the expression $w(\lambda) = \sum_{\mu \in N(\lambda)} \left(i_x(\lambda, \mu)\, i_y(\lambda, \mu) \right)$ as the criterion for removing a label in line 21 of the algorithm. This simple and fast heuristic works well in practice.
- After removing a label, we have chosen to increase the temperature to a value T_{reset} to facilitate more decisive changes in the surrounding area of the removed label.

Since the outer loop of the algorithm is executed $1 + k$ times, where k is the number of labels our algorithm has to delete to eliminate all intersections, the running time depends to a great extent on the number of unplaceable labels. Most problem instances in our test suits of real world labeling problems do not contain many unplaceable labels. Therefore, our method performs well on these problems.

3 Computational Results

We implemented the algorithm using the JAVA programming language and ran it on the same benchmark sets already used in [8]. This enables us to use the optimal values to evaluate our new algorithm. The first set is derived from real world data giving the positions of ground water drill holes in Munich and can be found on the website of Alexander Wolff [13]. The numbers of labels in these problems are in the set $\{250, 500, 750, \ldots, 2750, 3000\}$ and there are 30 instances for each number of labels. The second benchmark set has been computed using the randomized problem generator described in [1]. For each number in $\{100, 150, \ldots, 950, 1000\}$, we have 25 problem instances.

Figure 5 shows the difference of the running times of the exact branch-and-cut approach from [8] (called B&C in the plot) and our force directed method (FDL) on the generated benchmark set. The x-axis shows the number of labels in the problem while the y-axis shows the average computation time over all instances with the corresponding number of labels.

For all problem instances with up to 850 labels, we only considered the instances where B&C found the optimal solution after half an hour. The instances with more than 850 labels could not be solved by B&C. The plot shows that FDL is faster by several orders of magnitude than B&C (note that the y-axis is logarithmic). Both algorithms where executed on a Pentium 4 with 2.8 GHz and 2GB main memory running Linux. Note also that FDL is a JAVA implementation while B&C is written in C^{++}.

Figure 6 shows for the instances that B&C was able to solve in the given time the average quality of the solutions computed by FDL. We define the quality as the number of labels FDL could place divided by the number of labels in the optimal solution computed by B&C multiplied by 100. Even for the largest instances B&C was able to solve, the quality was over 98%.

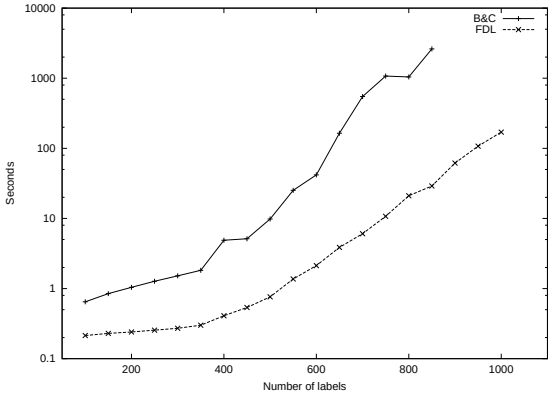


Fig. 5. Computation times of B&C and FDL for the generated benchmark set

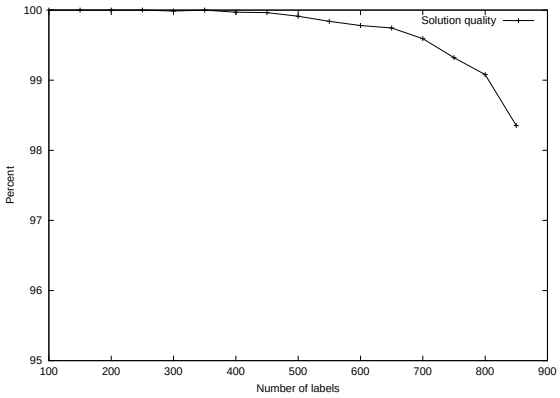


Fig. 6. Solution quality of FDL for the generated benchmark set

Figure 7 shows the running time of FDL on the benchmark set derived from the Munich drill holes. For these problems, the running time grows slowly and is on average less than 3.5 seconds for instances with 3000 labels.

To show that our new approach produces labelings with better label distribution than an algorithm like B&C that only optimizes the number of labels placed, we computed histograms that show the number of labels having a certain distance to their closest neighboring label as columns. Figure 8 shows the histograms for the generated benchmark set. We cut the x-axis at 60 because the number of labels with greater distance to their nearest neighbor was negligible.

Figure 8(a) shows the distribution of labels for B&C. Over a quarter of the labels have at most two points distance to their nearest labels. As Figure 8(b) shows, the situation is much better in the labelings produced by our new algorithm. It places only half as many labels at a distance of at most 2 Points than B&C and the distribution of the labels is more even. Figure 8(c) shows that using FDL as a post-processing tool on solutions computed by B&C achieves a similar distribution as FDL alone. Using this approach, we can have a labeling with the optimal number of labels placed together with an even distribution of the labels.

Figure 9 shows a labeling computed by FDL for a problem with 750 labels. Only four labels could not be placed (visible as outlines) and the computation was finished after five seconds. The distribution of the labels is very good and labels touch each other only in very dense areas.

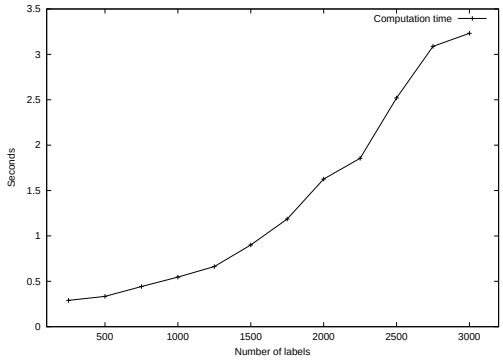
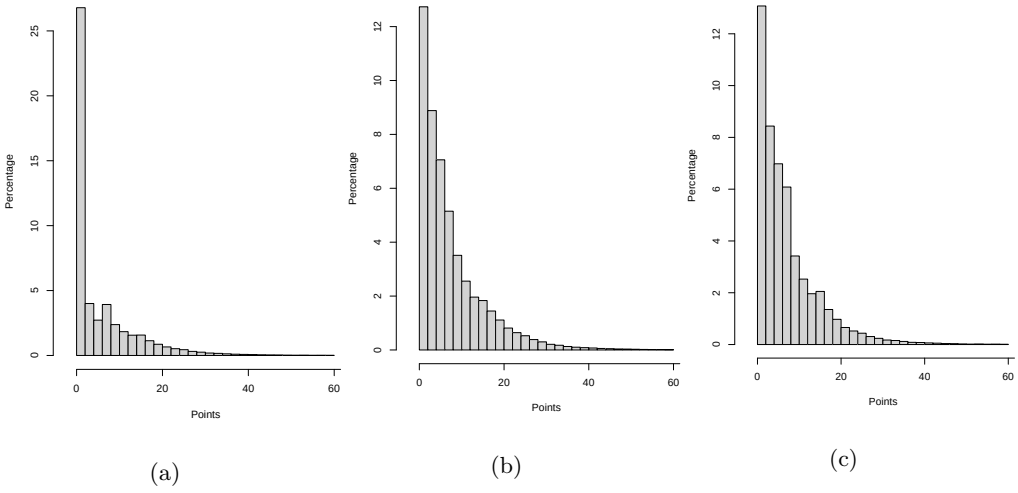


Fig. 7. Running time of our algorithm on the Munich drill holes benchmark set



(a) (b) (c)

Fig. 8. Histograms showing the distribution of the labels according to the distance to the closest neighboring labels for B&C (a), FDL (b), and B&C with FDL as a postprocessor (c)

4 Conclusions

We have presented a new hybrid heuristical approach for the label number maximization problem. Our algorithm uses an underlying force system that serves two purposes. First, a minimum energy configuration of this system corresponds to placements with evenly distributed labels that is appealing to a human observer. The second task of the force system is to determine which labels should be left out to obtain a labeling without overlaps. We combine this with a simulated annealing algorithm to escape local minima.

Our extensive computational experiments on widely used benchmark data show that our algorithm finds labelings that are close to optimality in a short amount of computing time. We find that our results often look similar to those of a human cartographer.

Future lines of research include to adapt the approach to line and area labeling and to related labeling problems such as the label size maximization problem. We also want to integrate the approach into the Human-Guided Search (HuGS) system, see [7], to allow for human interaction.

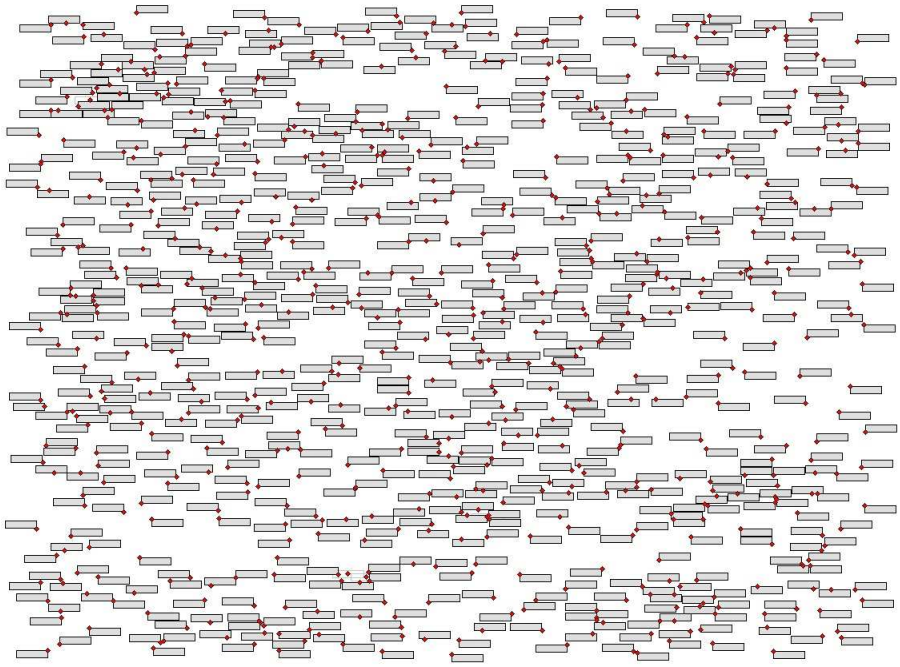


Fig. 9. Labeling placing 746 of 750 labels computed in less than 5 seconds.

References

1. 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.
2. R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics*, 15(4):301–331, 1996.
3. P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.
4. M. Feigenbaum. Method and apparatus for automatically generating symbol images against a background image without collision utilizing distance-dependent attrictive and repulsive forces in a computer simulation. United States Patent 5,355,314, 1993.
5. E. Imhof. Die Anordnung der Namen in der Karte. *International Yearbook of Cartography*, 2:93–129, 1962.
6. S. Kirkpatrick, Jr. C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.
7. G. W. Klau, N. Lesh, J. Marks, M. Mitzenmacher, and G. T. Schafer. The HuGS platform: A toolkit for interactive optimization. In *Proc. of AVI 2002 (International Working Conference on Advanced Visual Interfaces)*, 2002.
8. G. W. Klau and P. Mutzel. Optimal labelling of point features in rectangular labelling models. *Mathematical Programming*, 94(2-3):435–458, 2003.
9. J. Kruskal and J. Seery. Designing network diagrams. *First General Conf. on Social Graphics*, pages 22–50, 1980.
10. J. Marks and S. Shieber. The computational complexity of cartographic label placement. Technical Report TR-05-91, Harvard University, Cambridge, MA, U.S.A., 1991.
11. T. Strijk and M. van Kreveld. Practical extensions of point labeling in the slider model. In *Proc. 7th ACM Symp. Adv. Geogr. Inform. Syst.*, pages 47–52, 1999.
12. M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels. *Computational Geometry: Theory and Applications*, 13:21–47, 1999.
13. A. Wolff. The map labeling bibliography, 2003. URL location, google: "map labeling bibliography".
14. P. Yoeli. The logic of automated map lettering. *The Cartographic Journal*, 9:99–108, 1972.