Label Number Maximization in the Slider Model
Extended Abstract

Dietmar Ebner, Gunnar W. Klau, and René Weiskircher

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

Abstract. We consider 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 so that its boundary touches the corresponding point feature. The purpose of this paper is twofold: We present a new force-based simulated annealing algorithm to heuristically solve the problem and we provide the results of a very thorough experimental comparison of the best known labeling methods on widely used benchmark sets. The design of our new method has been guided by the goal to produce labelings that are similar to the results of an experienced human performing the same task. So we are not only looking for a labeling where the number of labels placed is high but also where the distribution of the placed labels is good.
Our experimental results show that the new algorithm outperforms the other methods in terms of quality while still being reasonably fast and confirm that the simulated annealing method is well-suited for map labeling problems.

1 Introduction

The growing amount of data for which informational graphics have to be produced leads to an increasing need for automatic labeling procedures.

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

(C1) On a good map the placement of labels is unambiguous. This implies that labels are close to the point features they belong to.
(C2) The information of the labels is legible.
(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 [6] and Yoeli [14]. Yet, the overall aim in automatic map labeling is to devise algorithms that produce labelings of maximum legibility.

J. Pach (Ed.): GD 2004, LNCS 3383, pp. 144–154, 2004.

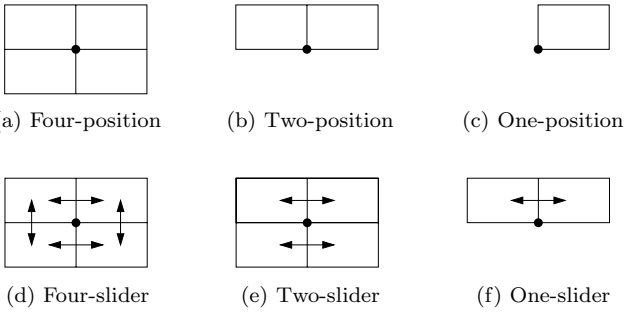


Fig. 1. Axis-parallel rectangular labeling models. A label can be placed in any of the positions indicated by the rectangles and can slide in the directions of the arrows.

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 (*label size maximization*), and keeping the sizes of the labels fix while looking for the maximum number of labels that can be placed (*label number maximization problem*, LNM).

Research in automated map labeling has mainly focused on the six *labeling models* shown in Figure 1, 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.

Definition 1 (LNM in the 4-slider model). *Given a set* $\Lambda = \{\lambda_1, \ldots, \lambda_k\}$ *(the labels), two functions* $w, h : L \to \mathbb{R}$*, and a function* $a : \Lambda \to \mathbb{R}^2$*, find a subset* $\Lambda' \subseteq \Lambda$ *of largest cardinality and a function* $\rho : \Lambda' \to \mathbf{R}$*, 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* $\lambda \in \Lambda'$*.*
(L2) Point $a(\lambda)$ *lies on the boundary of* $\rho(\lambda)$ *for all* $\lambda \in \Lambda'$*.*
(L3) The open intersection $\rho(\lambda) \cap \rho(\mu)$ *is empty for all* $\lambda, \mu \in \Lambda'$*,* $\lambda \neq \mu$*.*

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 in the 4-slider model. Property (L3) forbids overlaps between the labels.

Force-directed methods have originally been developed for drawing graphs. 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 [10], 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. 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.

Van Kreveld, Strijk, and Wolff [13] show *NP*-completeness of the decision problem in the four-slider model (independently, Marks and Shieber have shown this in [11]). The main result in [13] 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 $\Theta(n \log n)$-time greedy sweep-line algorithm. For the same models, the authors develop a polynomial time approximation scheme. Strijk and van Kreveld extend the above mentioned $\frac{1}{2}$-approximation algorithm for the slider models in [12] to labels with different heights. If r denotes the number of different label heights, the running time of the algorithm is $O(rn \log n)$. The algorithm is based on the simple greedy strategy of iteratively placing the *leftmost label* until no more points can be labeled without intersections. The *leftmost label* is defined to be the label, whose right edge is leftmost among all label candidates, which are those labels that have not been placed yet minus a set of labels that are already known to be unplaceable in the current configuration.

Klau and Mutzel present in [9] 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 combinatorial description 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 reasonably 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.

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 initial 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. 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.

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 of 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. To avoid being trapped in local minima of the energy function, we combine the purely force directed method with the simulated annealing approach.

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 labels are bound to their point feature and may not be positioned freely in the available space. We only allow intermediate positions that satisfy at least the first condition, hence we do not need any attractive forces between points and labels. Furthermore 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*

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

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. If labels overlap a very small area ϵ the intersection-proportional component can become arbitrary

small. Thus we add a constant value to the force function if and only if two labels overlap to punish overlaps stronger. 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.

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 $\mathbb{R}^2$.

We can now define the force function $f = (f_x, f_y)$ for each label in the following way: For each label $\lambda \in \Lambda$ with center point $c_\lambda = (x_\lambda, y_\lambda)$, the x-component of the force function $f_x : \Lambda \to \mathbb{R}$ is defined as

$$f_x(\lambda) = \sum_{\mu \in N(\lambda)} \left(f_i(\lambda, \mu) + f_a(\lambda, \mu) + f_d(\lambda, \mu)\right) (x_\lambda - x_\mu) / (\|c_\lambda, c_\mu\|) \ ,$$

$$\text{where} \qquad f_i(\lambda, \mu) = \delta_1 \, i_x(\lambda, \mu) \, i_y(\lambda, \mu) \ ,$$

$$f_d(\lambda, \mu) = \frac{\delta_2}{\max(\varepsilon, d_{\min}(\lambda, \mu))^2} \ , \quad \text{and} \quad f_a(\lambda, \mu) = \begin{cases} \delta_3 & \text{if } \lambda \text{ and } \mu \text{ overlap} \\ 0 & \text{otherwise .} \end{cases}$$

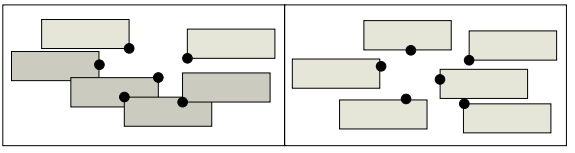
The *y-component* f_y is defined analogously. The constants $\delta_1, \delta_2, \delta_3 \in \mathbb{R}$ control the influence of the particular term on the force function f. Note that the direction of the force between two labels is defined by the location of their center points and that ε limits the amount of f_d to a value of δ_2/ε^2.

"Force has no place where there is need of skill." [4]

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. Often, real-world labeling instances contain dense areas that 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. The same observation holds for the algorithm proposed by Hirsch in [5] and is well described by Christensen, Marks, and Shieber in [1].

Figure 2(a) 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 continuously 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 the forces. 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 2(b) 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, even though the optimal solution is self-evident.



(a) Bad local minimum vs. optimal labeling.

(b) Forces cannot alwyas resolve overlaps.

Fig. 2. Problems with forces.

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.

Simulated annealing is a very flexible optimization method and can be used in a wide range of combinatorial optimization problems. It has been proposed in [7] and 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.

The hybrid force-based simulated annealing algorithm for the label number maximization problem works as follows:

1: compute random initial labeling σ in the eight-pos. model
2: initialize temperature T and cooling rate α
3: compute forces for current conf. and init. set of active labels Φ
4: $M_{\max} \leftarrow 30|\Lambda|$; taken $\leftarrow$ rejected $\leftarrow 0$;
5: **repeat**
6: $\quad\hat{\sigma} \leftarrow \sigma$;
7: $\quad$choose random candidate $\lambda \in \Phi$
8: $\quad$**if** $|f_x(\lambda)| > F_{\min} \vee |f_y(\lambda)| > F_{\min}$ **then**
9: $\quad\quad$change σ by moving λ in the direction indicated by its force vector
10: $\quad\quad$**if** $|f_x(\lambda)| > F_{\min} \vee |f_y(\lambda)| > F_{\min}$ **then**
11: $\quad\quad\quad$change σ by moving λ to a random position
12: $\quad$**else**
13: $\quad\quad$change σ by moving λ to a random position
14: $\quad$**if** force$(\sigma) <$ force$(\hat{\sigma}) \vee$ random $r \in [0 \ldots 1] < e^{\frac{\text{force}(\sigma)-\text{force}(\hat{\sigma})}{T}}$ **then**
15: $\quad\quad$taken $\leftarrow$ taken $+ 1$
16: $\quad\quad$update set of active labels Φ
17: $\quad$**else**
18: $\quad\quad$rejected $\leftarrow$ rejected $+ 1$
19: $\quad\quad\sigma \leftarrow \hat{\sigma}$;

20: **if** taken + rejected $\geq M_{max}$ **then**
21: **if** taken $= 0$ **then**
22: **if** point selection is disabled $\vee \neg\exists$ overlapping label $\hat{\lambda} \in \Phi$ **then**
23: **return** current labeling σ
24: **else**
25: $\sigma \leftarrow \sigma \setminus \hat{\lambda}$
26: update set of active labels Φ
27: $T = \alpha T$
28: $M_{\max} = \max(|\Lambda|, \min(10|\Lambda|, 50|\Phi|))$; taken $\leftarrow$ rejected $\leftarrow 0$
29: **until** $|\Phi| = 0$
30: **return** current labeling σ

The algorithm performs a series of temperature stages. After each stage the temperature is decreased by a constant precomputed factor, which decreases the probability of accepting moves that lead to a higher energy state. To speed up convergence we compute a set of *active labels* Φ, which either intersect at least one other label or their associated force vector indicates movement to a new position with lower energy. The algorithm returns the current solution if $|\Phi| = 0$ and chooses the label with the most overlaps for removal if no move has been accepted for a full temperature stage.

In each iteration we randomly choose a label $\lambda \in \Phi$ and try to move it according to its force vector. If this move does not lead to an equilibrium or the force vector does not indicate movement even though the label is involved in an overlap, we move the label to a random position in the eight-position model instead. 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.

At each temperature stage we perform $M_{\max}$ moves. We initialize this value with $30|\Lambda|$ and perform $\max(|\Lambda|, \min(10|\Lambda|, 50|\Phi|))$ moves in all subsequent stages. The initial temperature is chosen such that we accept an increase in the overall force of f_{avg} with a probability of 30%, where f_{avg} represents the amount of force for an overlap of 50% of two average sized labels. The cooling rate α is chosen such that the temperature T becomes less than 1 after 15 stages. The parameter α should be changed to adjust the trade-off between quality and speed. The above settings yield high-quality labelings in reasonable computation time.

Whenever we move a label λ to a different position or remove it from the labeling in line 25, the forces on all labels $\lambda' \in N(\lambda)$ change. Since a simple approach takes time $O(n^2)$ in the worst case we store the forces between each pair of labels in a quadratic matrix. This enables us to update the forces in linear time by recalculating only the change of the particular addend for each neighbor $N(\lambda)$. Furthermore we have to update the set of active labels Φ, since some labels $\lambda' \in N(\lambda)$ may have to be added to or removed from this set.

Since labels have to be placed according to the four-slider model, moving a label alongside its force vector becomes more difficult than moving, *e.g.*, zero-sized nodes in a graph drawing application. A position that corresponds to an

equilibrium of the forces is not always valid with respect to the point. Furthermore 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. Thus we can not apply numerical algorithms like the Newton-Raphson method or similar techniques, since they require at least the first derivation of the function. In place of this we start moving the label by 20% of the remaining width/height in the particular direction and halve the amount of movement if the indicated direction changes until we achieve an equilibrium or the maximum number of moves has been performed.

We perform at least $|\Lambda|$ moves before removing a heuristically chosen label. Thus the running time depends to a great extent on the number of labels that the algorithm cannot place. Most problem instances in our test suits of real world labeling problems do not contain many of these unplaceable labels. Therefore, our method performs well on these problems. However, if running time is a critical criterion, this step can be replaced through a faster cleanup heuristic.

3 Computational Study

In this section we report on the extensive computational experiments we have performed to evaluate quality and resource requirements of our new method in comparison to the best-known algorithms for label number maximization. We want to emphasize that both the data we used and our implementation of the evaluated algorithms are publicly available under the Gnu General Public License at `http://www.ads.tuwien.ac.at/research/labeling`.

We have implemented all major map labeling algorithms that we found in the literature on point feature map labeling in the slider model. All computations were done on a Pentium 4 with 2.8GHz and 2GB of RAM. For each run, we set a limit of 30 minutes computation time.

- The algorithm RANDOM, which places labels randomly, if possible, has been incorporated into the study only for comparative reasons.
- Christensen, Marks and Shieber present in [1] a simulated annealing approach that beats most other algorithms in both speed and quality. Since their implementation uses the four-position model, in general, the quality of their solutions cannot be as good as those of algorithms for the four-slider model. Nevertheless we decided to include this algorithm in our computational study to compare one of the best known labeling methods in the four-position model to the remaining algorithms. We isolated configuration changes to either obstructed or deleted labels, since this causes the algorithm to converge much faster.
- We followed the suggestion of Christensen, Marks, and Shieber in [1] and reduced the radius of the circle in Hirsch's algorithm (HIRSCH) to zero. Furthermore we neglect any cartographic preferences.
- APPROX is our implementation of the algorithm described in [13, 12] that runs in $O(n^2)$, does not rely on unique label heights, and is quite fast in practice.

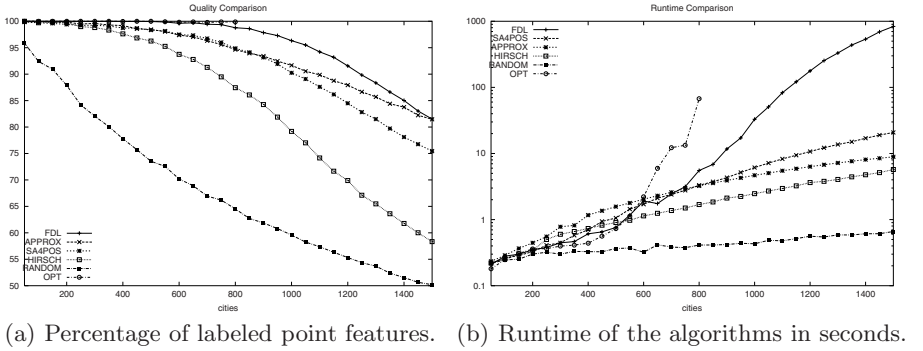
– We computed optimal labelings in the 4-slider model using OPT, an implementation of the algorithm presented in [9]. Note that, due to the running time limit, only instances up to maximally 850 labels could be computed.
– Finally, FDL is our JAVA implementation of the new force-directed method.

We ran the implementations on different data sets. Among them are (a) instances generated with the widely used benchmark generator by Christensen, Marks, and Shieber and (b) instances derived from real world data giving the positions of ground water drill holes in Munich.

We generated 25 random problem instances of type (a) for each instance size in $\{100, 150, \ldots, 1450, 1500\}$ labels, resulting in 685 instances, as in the study on the four-position model [1]. The numbers of labels in the real-world problem set (b) are in the set $\{250, 500, 750, \ldots, 2750, 3000\}$ and there are 30 instances for each number of labels.

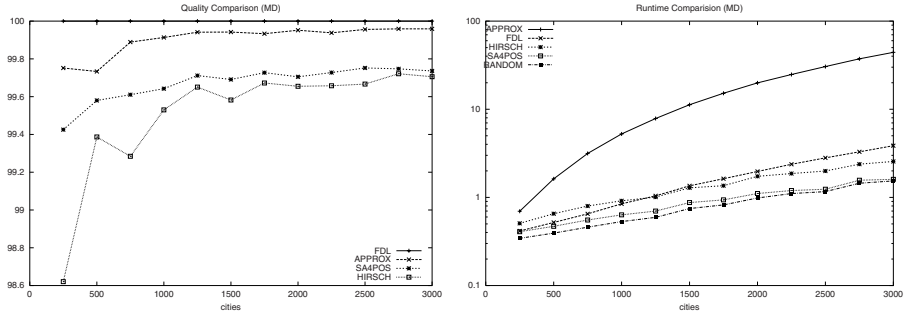
Figure 3(a) illustrates the performance of the evaluated algorithms in terms of quality, whereas Figure 3(b) displays their running time behavior. Of course, OPT performs best in terms of quality but also needs the largest amount of resources. Among the heuristic methods, our new algorithm produces the best scores but also takes more time to compute them – especially for large instances. We want to remark, however, that the random instances larger than 1000 labels do not resemble real-world instances since they get very dense (see the discussion on the real-world Munich drill hole instances below). The plots also reveal that the approximation algorithm performs surprisingly well in terms of quality (for very large instances it becomes as good as FDL) with the advantage that its running time does not explode.

We then compared the heuristic methods on the easier real-world instances. Figures 4(a) and 4(b) show the results. It can be seen that all methods apart from RANDOM have quite good results with FDL being the winner. In fact, these instances have been generated so that always 100% of the labels can be placed – even in the four-position model. FDL is the only method that achieved the perfect score on all instances. As already mentioned, the running time of FDL depends heavily on the number of labels that cannot be placed. As this number



(a) Percentage of labeled point features. (b) Runtime of the algorithms in seconds.

Fig. 3. Results for the random benchmark set.



(a) Percentage of labeled point features (RANDOM is far behind and thus not shown).

(b) Runtime of the algorithms in seconds.

Fig. 4. Results for the real-world benchmark set.

is zero for these instances, the running time behavior is very good for FDL as for all other methods apart from the approximation algorithm.

Our computational results confirm the outcome of the 1995 study [1]: simulated annealing is very well-suited for labeling problems and outperforms other methods in terms of quality.

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 might include to adapt the approach to line and area labeling. We will also investigate how to combine force-based graph drawing with our approach to attack the combined drawing and labeling problem. Further, we want to integrate the approach into the Human-Guided Search (HuGS) system, see [8], to allow for human interaction.

Acknowledgments

We thank Tycho Strijk for useful discussions regarding the implementation APPROX and one of the anonymous referees of a previous version for his constructive feedback.

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. Herodotus. *The History of Herodotus*. 440 B.C.
5. S. A. Hirsch. An algorithm for automatic name placement around point data. *The American Cartographer*, 9:5–17, 1982.
6. E. Imhof. Die Anordnung der Namen in der Karte. *International Yearbook of Cartography*, 2:93–129, 1962.
7. S. Kirkpatrick, Jr. C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.
8. 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.
9. G. W. Klau and P. Mutzel. Optimal labelling of point features in rectangular labelling models. *Mathematical Programming*, 94(2-3):435–458, 2003.
10. J. Kruskal and J. Seery. Designing network diagrams. *First General Conf. on Social Graphics*, pages 22–50, 1980.
11. 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.
12. 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.
13. M. van Kreveld, T. Strijk, and A. Wolff. Point labeling with sliding labels. *Computational Geometry: Theory and Applications*, 13:21–47, 1999.
14. P. Yoeli. The logic of automated map lettering. *The Cartographic Journal*, 9:99–108, 1972.