An efficient solution to Knuth's METAFONT labeling problem*

Michael Formann[§] Frank Wagner[§]

Abstract

In a recent paper Knuth and Raghunathan showed that the so called METAFONT problem is NP-complete. In this problem one has to attach square 2×2 labels immediately to the right, left, above or below given sites on the $\mathbb{Z} \times \mathbb{Z}$ grid, such that all squares are disjoint.

The importance of MFL and similar problems goes far beyond the application in METAFONT, a program for font design, itself. It is a problem cartographers are often faced with, when they assign labels (e.g. sulphate-concentrations) to geographic points (e.g. groundwater-drillholes). To be useful in cartographic applications the restriction to 2×2 squares is too strict. We therefore look at the following associated more general optimization problem: Given n sites in the plane, determine the maximum value σ_{max}, such that for each point there is one square with lateral length σ_{max} attached immediately to the right, left, above or below to it. These squares are not allowed to overlap.

We show that we can approximate σ_{max} by at least $33\frac{1}{3}\%$ in $\mathcal{O}(n \log n)$ time.

Keywords compatibility, approximation algorithm, cartography, NP-complete

AMS (MOS) subject classifications 68Q25 (Analysis of algorithms and problem complexity); 65Y25 (Computer graphics and computational geometry)

Abbreviated Title The METAFONT labeling problem

*This work was partially supported by the ESPRIT II Basic Research Action of the European Community under contract No. 3075 (project ALCOM).

§Institut für Informatik, Fachbereich Mathematik, Freie Universität Berlin, Arnimallee 2–6, W1000 Berlin 33, Germany; e-mail: `formann@tcs.fu-berlin.de, wagner@tcs.fu-berlin.de`

1 Introduction

In a recent paper Knuth and Raghunathan [4] introduce the name *the problem of compatible representatives* for a class of combinatorial problems. Many combinatorial tasks can be formulated as follows:

Is there a sequence $(x_1, x_2, \ldots, x_n)$ such that $x_j \in A_j$ and x_j is compatible with x_k for all $j \neq k$? Here $A_1, A_2, \ldots, A_n$ are given "candidate sets" and "compatibility" is a given relation on $A_1 \cup A_2 \cup \ldots \cup A_n$. In general, this problem is NP-hard. Various special cases, where the problem is efficiently solvable are presented in [4].

As a hard case Knuth and Raghunathan show that the so called MFL METAFONT labeling) problem is NP-complete. This problem is formulated in the following way: Given a set $p_1, p_2, \ldots, p_n$ of integer points (sites) in the plane, find integer points $x_1, x_2, \ldots, x_n$ such that

$$\begin{aligned} |x_j - p_j| &= 1 && (1)\\ |x_j - p_k| &> 1 && (2)\\ |x_j - x_k| &\geq 2 && (3) \end{aligned}$$

for all $1 \leq j, k \leq n$, $j \neq k$. Note that here $|\cdot|$ denotes the Euclidean L_2 metric. The conditions (1)–(3) can be interpreted as follows: Each x_j is the center of a 2×2 square, which is the bounding box of some "label" attached to its site. Condition (1) means that each label is immediately to the right, to the left, above or below its site. Condition (2) prevents a label from being placed next to another site (on a printed map it should always be possible to determine uniquely and easily which label belongs to a site and vice versa) and condition (3) prevents two labels from overlapping and therefore being illegible.

The importance of MFL and similar problems goes far beyond the application in METAFONT itself. Indeed it is a problem cartographers are often faced with. To be useful in cartographic applications (cf. [3]) the restriction to 2×2 squares is too strict. We therefore look at the following more general problem:

> CARTO-DECISION
> INSTANCE: Set $p_1, p_2, \ldots, p_n \subseteq R^2$ of points (sites) in the plane and a common label size $\sigma \in R$.
> QUESTION: Are there axis-parallel squares $S_1, S_2, \ldots, S_n$ (labels) of lateral length σ, such that
>
> (1) each square S_j has its site p_j exactly in the middle of one of its sides,
> (2) no other point p_k $(k \neq j)$ lies in the middle of one of S_j's sides and
> (3) no two squares S_j and S_k intersect?

Clearly MFL is a special case of CARTO-DECISION (with σ set to 2) and therefore CARTO-DECISION is also NP-complete and — most likely — not solvable in polynomial time. Anyhow, a negative answer for a specific instance would not be very helpful for the mapmaker. A way out of this dilemma is to reduce the label size.

This idea was applied by Formann and Wagner [2] for a different way of labeling points. In the main part of this note we will show that the methods developed there also apply for MFL resp. CARTO-DECISION.

2 The Approximation Algorithm

The informal discussion at the end of the preceding section leads to the following problem CARTO-OPTIMIZATION which is the optimization version of CARTO-DECISION.

> CARTO-OPTIMIZATION
> INSTANCE: Set $p_1, p_2, \ldots, p_n \subseteq R^2$ of points (sites) in the plane.
> QUESTION: What is the maximum σ_{max} of all reals σ such that there are axis-parallel squares $S_1, S_2, \ldots, S_n$ (labels) of lateral length σ, with
>
> (1) each square S_j has its site p_j exactly in the middle of one of its sides,
> (2) no other point p_k $(k \neq j)$ lies in the middle of one of S_j's sides and
> (3) no two squares S_j and S_k intersect?

Because CARTO-DECISION is NP-complete all we can hope for is an efficient approximation algorithm for CARTO-OPTIMIZATION. For that reason, we will show that methods developed in [2] carry over. Our main result is that σ_{max} can be approximated with at least $33\frac{1}{3}\%$.

The approximation algorithm, to be presented here, conceptually works as follows: We start with infinitesimal equally sized squares, in all four possible positions, attached to each point. Then all squares are expanded uniformly. Eventually squares belonging to different points start overlapping. In this case an action has to be performed. Normally one or both squares are eliminated and no longer candidates for a solution. In the other case the two squares are coupled by means of a logic formula. This process continues as long as there is still a solution.

To be precise we give the following definition:

Definition 1 *For a point p in the plane denote by p^i $(i \in \{1,2,3,4\})$ an axis-parallel unit square with p in the middle of its left, lower, right, resp. upper side; for a real σ denote by σp^i analogously a square with lateral length σ.*
For a point set $P \subseteq R^2$ (sites) and a real σ we call p^i $(p \in P)$

*(1) σ-*dead*, if $3\sigma p^i$ contains another point $q \in P$ $(p \neq q)$;*

*(2) σ-*pending*, if p^i is not σ-dead and σp^i has nonempty intersection with σq^j, where q^j $(p \neq q)$ is another non-σ-dead square;*

*(3) σ-*alive*, if p^i is neither σ-dead nor σ-pending.*

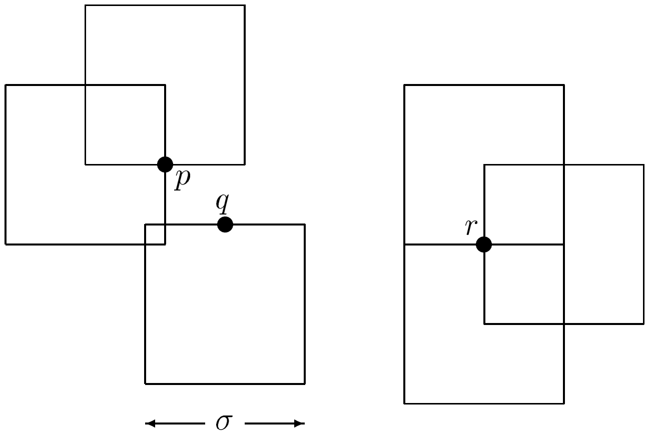

Figure 1: 3 points; p^3 and q^4 are σ-pending; p^2,r^1,r^3,r^4 are σ alive; σ-dead squares are not shown.

Figure 1 illustrates this definition. With this definition we are enabled to characterize the situation for some fixed σ. Let $P \subseteq R^2$ be a set of sites in the plane. The following facts are easy to proof:

Fact 1 *No σ-alive square intersects with another square from a different site.*

Only σ-pending (and σ-dead which we are not interested in) squares are allowed to intersect (with other σ-pending squares).

Fact 2 *Each site has at most two σ-pending squares.*

Proof:
Suppose two pending squares intersect, say p^1 and q^j, then $j = 2$ or $j = 4$. W.l.o.g. $j = 2$. But then p^3 and p^4 are σ-dead.

By Fact 1 any σ-alive square can be used as a legal label for its point. But σ-pending squares do intersect with σ-pending squares of other sites. So we have to take precautions if a point has only σ-pending squares left. We call such points *σ-pending points.* Fact 2 enables us to express the intersection structure of pending squares with a simple logic formula. We consider the sites as Boolean variables. The (at most) two σ-pending squares will be associated with the two logic states. If two σ-pending squares, say p^i (associated with the logic state p) and q^j (associated with $\overline{q}$) intersect, then we form the clause

$$\overline{(p \wedge \overline{q})} = \overline{p} \vee q.$$

This gives us a set of clauses that has the following property:

Fact 3 *There is a solution for the set of σ-pending points if and only if the corresponding set of clauses is satisfiable.*

Furthermore the clauses have a relative simple structure, each of them has only two literals. To find a satisfying truth assignment is an easy task, it is the polynomial (linear in the number of clauses) **2-SAT** problem (cf. [1]).

So far we have two possibilities that there is no longer a solution. Either there are points that have no squares left or the set of clauses for σ-pending points does not have a satisfying truth assignment. In both of these cases the algorithm stops. In an implementation of the algorithm the continous expanding of σ is simulated by discrete steps. We are interested only in those values of σ for which a square changes its state. These $\mathcal{O}(n)$ "events" can be found in $\mathcal{O}(n \log n)$ time by known techniques (e.g. plane-sweep [5, p. 147–160]) of computational geometry.

We are now ready to present the algorithm:

Approximation Algorithm:

1 *compute all events;*
2 $\sigma := 0$*;*
3 *assign the state σ-alive to all four squares of each point;*
4 **while**
5   *all points have non-σ-dead squares* **and**
6   *the set of clauses for the σ-pending points is satisfiable*
7 **do begin**
8   *increase σ to the next event;*
9   *update the states of all squares;*
10 **end**;
11 *undo the last state update;*
12 {*construct a solution:*}
13   *choose some fixed truth assignment Φ for the set of clauses for σ-pending squares;*
14   *for each non-σ-pending-point choose one of its σ-living squares;*
15   *for each σ-pending-point choose one of its σ-pending squares according to Φ;*

Note, that in principle, the state of a square should change *immediately after* rather than *at* an event. (The squares are considered to be topologically open.) This is the reason why step 11 makes sense.

Let us look at the performance of the algorithm.

Theorem 1 *The Approximation Algorithm yields a solution with at least $33\frac{1}{3}\%$ of the optimal size.*

Proof

Look at some solution of optimal size σ_{max}. Obviously it can be shrinked to a solution for size $\sigma_{max}/3$. Those, loosely speaking, "$\frac{1}{3}$-squares" are certainly not $\sigma_{max}/3$-dead. So, supplied at least with these $\frac{1}{3}$-squares, our algorithm has not stopped before it reached $\sigma_{max}/3$.

As already said, step 1 of our algorithm can be implemented in $\mathcal{O}(n \log n)$ time. This run-time is also possible for the rest of the algorithm, all we have to discuss further is how to perform step 6. The clause structure behaves very locally. To be satisfiable any variable must occur only in at most two clauses with at most one other variable. Now it should be clear that step 6 can be performed in constant time and we have:

Theorem 2 *The Approximation Algorithm can be implemented in* $\mathcal{O}(n \log n)$ *time.*

Clearly, be rescaling one of the coordinates, our algorithm can also be applied to equally-sized rectangles. Furthermore, by using affine transformations, our algorithm can be used for such strange label shapes as parallelograms.

References

[1] S. Even, A. Itai and A. Shamir, *On the complexity of timetable and multi-commodity flow problems*, SIAM J. Comput. **5** (1976), pp. 691–703

[2] M.Formann, F.Wagner, *A Packing Problem with Applications to Lettering of Maps*, Proceedings of the 7th ACM Symposium on Computational Geometry (1991), pp. 281–288

[3] E. Imhof, *Positioning Names on Maps*, The American Cartographer **2** (1975), pp. 128-144

[4] D.E. Knuth and A. Raghunathan, *The Problem of Compatible Representatives*, SIAM J. Disc. Math. **5** (1992), pp. 422–427

[5] K. Mehlhorn, *Data structures and Algorithms, vol. 3: Multi-Dimensional Searching and Computational Geometry*, Springer Verlag, Berlin, Germany, 1984