アルゴリズム 73－2
（2000．5．19）

可変形状ラベリング問題に対するアルゴリズム

中野　眞一 (群馬大学)、西関　隆夫、徳山　豪、渡部修平 (東北大学)

要旨. 障害物のある平面上の点集合に対するラベル付け問題で、LOFL(左整列可変ラベリング）という手法を提案する。各点に対して, 左整列性を持つ候補ラベル集合が与えられ, ラベルは候補から選ばれて共通のスケール比で拡大/縮小されて配置される。本論文では，固定スケール比の入力に対して配置可能決定に対する最適な $O((n+m)\log(n+m))$ 時間アルゴリズムと最大スケール比を求める $O((n+m)\log^2(n+m))$ 時間のアルゴリズムを与える．ここで n はラベル付けされる点の数であり、m は障害物の複雑度である。LOFL では過去のラベリング手法と違い、様々な形のラベルを混用できる。そのため、図 1 にあるように、スケール比の大きなラベル付けを可能にする。

Algorithms for Shape-Flexible Labeling of Points

Shin-ichi Nakano[†],, Takao Nishizeki[‡], Takeshi Tokuyama[‡], Shuhei Watanabe[‡]

[†]Department of Information Engineering, Gunma University, Kiryu 376-8515, Japan, Email: nakano@cs.gunma-u.ac.jp

[‡]Graduate School of Information Sciences, Tohoku University, Aobayama, Sendai 980-8579, Japan. Emails: (nabe,nishi)@nishizeki.ecei.tohoku.ac.jp, tokuyama@dais.is.tohoku.ac.jp

Abstract. We deal with a map labeling problem, named LOFL (Leftside-Orderly Flexible Labeling), to label a set of points in a plane with polygonal obstacles. The label for each point is selected from a set of rectangles satisfying the *leftside-orderly* property and is placed near to the point after scaled by a scaling factor σ which is common to all points. In this paper, we give an optimal $O((n+m)\log(n+m))$ algorithm to decide the feasibility of LOFL for a fixed scaling factor σ, and an $O((n+m)\log^2(n+m))$ time algorithm to find the largest feasible scaling factor σ, where n is the number of points and m is the number of edges of polygonal obstacles. Figure 1 illustrate the improvement of the scaling factor achieved by LOFL, where we can use three different kinds of labels all together.

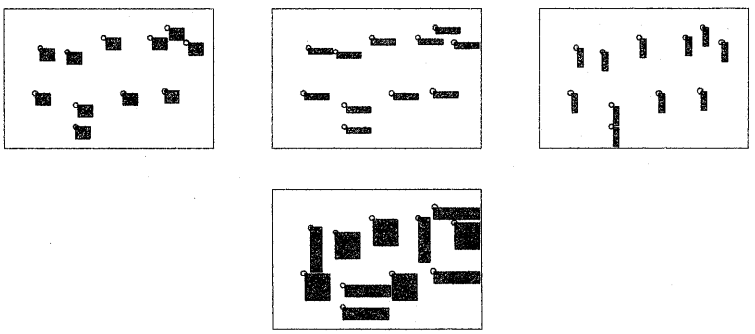

図 1: Placements using three kinds of labels.

1 Introduction

Annotating a set of points is a common task to be performed in Geographic Information Systems. It is crucial that important objects in a map have labels indicating their names or other attributes. The objects to be labeled in a map highly depend on user's interest; for example, a drainage maintainer may want to have locations and identification labels of manholes, although they are almost useless information for ordinary users. Therefore, a digital map should have a database of sets of points representing locations of objects together with labels of the objects, and should have a function to insert labels to a non-labeled map efficiently.

The problem of locating labels in a map is called the map labeling (or map lettering) [11, 16, 17, 18]. Approximating a label (a string of characters) by its bounding rectangle, one can formulate the map-labeling problem as the problem of locating a set of n rectangles in a plane (with obstacles containing m edges) in a way that (1) each rectangle representing a label of an object should be near to the object, (2) rectangles do not overlap each other, and (3) each rectangle does not overlap any obstacle in the map. We allow a rectangle to touch other rectangles or obstacles on its boundary. The condition (1) will be mathematically formulated in a suitable fashion.

We restrict ourselves to the *point feature label placement problem*, where each object is a point (object point) in the map. The rightmost point of an object is often chosen as an object point. Moreover, we only consider axis parallel rectangles as labels. See [4, 9] for more complicated labeling problems.

If the size of each character is given (therefore, the size of each label is given), we want to decide whether there exists a feasible solution satisfying the conditions (1), (2), and (3) above. Such a problem is called the *decision problem*. We also want to consider the *optimization problem* in which we compute the maximum character size σ, called a *scaling factor*, for which there is a feasible solution.

The decision problem is hard in general: Formann and Wagner [7] showed that if there are four candidates of the placement for each label, it is NP-hard in general to decide the feasibility. Indeed, it is NP-hard even if each label is a unit square and must be placed in a way that the corresponding object point is at one of its four corners; we say that such a label is *pinned* at a corner. Kato [10] showed that the problem remains to be NP-hard if there are three candidates for each label.

On the other side, if each label is a unit square pinned at one of its two left corners, the problem is polynomial time solvable. In general, if there are at most two candidates of the placement for each label, the problem is polynomial-time solvable since it can be formulated as a 2-SAT problem [7]. Moreover, approximation algorithms with provable approximation ratios are given for several useful versions of the map labeling [11, 16]. If we fix the scaling factor and measure the quality of the solution with the number of labels that can be placed without overlapping, there are PTAS algorithms provided that the ratio of heights of the tallest label and the shortest label is bounded by a constant [11, 12], although their time complexities are rather high unless the intersection graph associated with candidate labels is very sparse.

We deal with another type of a map-labeling, called a *shape-flexible labeling*, where we can flexibly choose the shape of each label from a candidate set of rectangles. The chosen labels are placed in the map after scaled by a scaling factor σ which is common for all labels. The problem of deciding the feasiblity of a shape-flexible labeling problem is NP-hard in general, and the complexity of solving the problem heavily depends on features of candidate sets. If each candidate set is the set of all rectangles with a given area, the labeling is called an *elastic labeling*, and some special cases were in-

vestigated by Iturriaga and Lubiw [8].

In this paper, we propose a new class of shape-flexible labeling problems, named *Leftside-Orderly Flexible Labeling* (LOFL), where the candidate set of rectangles given to each object point must be a *leftside-orderly* set. The definition of a leftside-orderly set will be given in the next section; a typical example is a set of rectangles pinned at their left-upper corners.

Our motivation is as follows: For example, consider a label "Graduate School of Information Sciences." It needs width 39 (character units) if it is written in a single line. However, it can be written in two lines using width 20, or in four lines using width 11 without breaking words. Thus, we can fold a label to decrease the width. Moreover, in the Chinese (also Japanese or Korean) language system, we can write a character string vertically, and hence we can transpose a label (i.e. exchange its width and height). It is often seen that folding and transposition can improve the labeling layout. By formulating such a problem as LOFL, we can compute the optimal folding and transposition of labels to maximize the character unit.

We show that the decision problem for LOFL can be solved in $O((n+m)\log(n+m))$ time by a simple plane-sweep algorithm. As a consequence, the optimization problem for LOFL can be solved in $O((n+m)\log(n+m)\log\Gamma)$ time if the coordinate values of points are represented by $\log\Gamma$-bit integers. We also give an $O((n+m)\log^2(n+m))$ time algorithm for the optimization problem. To design this algorithm, we use the parametric searching paradigm in a novel way; we use parallel sorting and point location query to design a "guide algorithm" required for the parametric searching.

Our method can be used as a subroutine in heuristic algorithms for a practical labeling system. We give some experimental results to compare several labeling schemes related to LOFL.

2 Preliminaries

2.1 Leftside-orderly sets

A set $\mathcal{I}$ of intervals is *totally ordered* with respect to inclusion if any pair I and I' of intervals in $\mathcal{I}$ satisfies either $I \subset I'$ or $I' \subset I$.

For a rectangle L representing a label, we denote by $left(L)$ the left edge of the rectangle L. Also, we fix a point $q(L)$ on $left(L)$, which we call the *pinning point* of L. Consider a set $\mathcal{L}$ of rectangles with pinning points. Suppose that we locate the rectangles in $\mathcal{L}$ so that their pinning points are translated to the origin. Then, $left(\mathcal{L}) = \{left(L) | L \in \mathcal{L}\}$ is a set of intervals on y-axis. If $left(\mathcal{L})$ is totally ordered, we say the set $\mathcal{L}$ satisfies the *leftside-orderly property*, and call the set $\mathcal{L}$ a *leftside-orderly set*. See Figure 2 (a) for an example. In this paper, a leftside-orderly set always implies that a pinning point is given for each rectangle in the set.

We are given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n object points on the plane. Let $p_i = (x_i, y_j)$ for $i = 1, 2, \ldots, n$. We also consider a set Q of polygonal (not necessarily rectilinear) obstacles in the plane, which no label is permitted to overlap. We assume that the obstacles do not intersect each other. Let m be the number of edges of polygons in Q. A leftside-orderly set $\mathcal{L}_i$ of rectangular labels is given to each object point $p_i \in P$. $\mathcal{L}_i$ and $\mathcal{L}_j$ may be different from each other if $i \neq j$.

Let $N = \sum_{i=1}^{n} |\mathcal{L}_i|$: N is the number of all candidate rectangles. In Geographic Information System, the cardinality of a leftside-orderly set for a point is usually bounded by a constant. Therefore, we assume $N = O(n)$ in this paper, although it is not difficult to generalize our argument to the cases where N is much larger than n.

Let σ be a positive real value, called a *scaling factor*. We choose a suitable label from $\mathcal{L}_i$ for each $p_i \in P$, scale it by the factor σ, and place it in the plane so that its pinning point is located at p_i. The placement is called *feasible* if (1) no two

labels overlap each other and (2) no label overlaps any obstacle. Figure 2(b) illustrates a feasible placement for a set P of five points, to each of which the set in Fig. 2(a) is assigned. Our aim is to find the largest scaling factor σ for which a feasible placement exists, and to compute the placement.

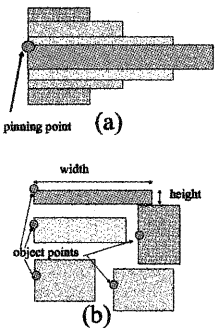

図 2: (a) A leftside-orderly set consisting of four rectangles, and (b) a labeling by LOFL.

Without loss of generality, we assume that there is no pair of labels L and L' in $\mathcal{L}_i$ such that $L \subset L'$, since we need not use the larger label L' in our solution if such a pair exists.

We define an ordering $>$ on the set $\mathcal{L}_i$ as follows: $L > L'$ for two labels L and L' in $\mathcal{L}_i$ if and only if $left(L) \supset left(L')$. We say that L' is *shorter* than L and *wider* than L if $L > L'$, since L' is shorter in height and wider in breadth than L because of the assumption above.

Using the ordering, we can naturally give a lexicographical ordering among the set of feasible solutions (for a fixed σ) as follows: We sort the object points $p_1, p_2, \ldots, p_n$ in nonincreasing order with respect to the x-coordinate values, and re-arrange the numbering; hence, p_1 is the rightmost point and p_n is the leftmost point. Let $\mathbf{L} = (L_1, L_2, ..., L_n)$ and $\mathbf{L'} = (L'_1, L'_2, ..., L'_n)$ be two different feasible placements where L_i and L'_i are labels for p_i. Then we define $\mathbf{L} > \mathbf{L'}$ if there is an index j such that $L_j > L'_j$ and $L_i = L'_i$ for every $i < j$. The minimum feasible placement with respect to this ordering is called the "widest" solution.

2.2 Approximation hardness for label sets without leftside-orderly property

Before presenting algorithms for LOFL, we remark that the leftside-orderly property is crucial for designing a polynomial time algorithm. Indeed, it is NP-hard to compute a feasible placement whose scaling factor is larger than $\frac{1}{2}+\epsilon$ times the optimal scaling factor for any positive constant ϵ. The hardness result can be easily obtained by modifying the reduction of the planar 3SAT problem to a labeling problem with three candidate labels [10], and hence the proof is omitted in this version. On the other hand, one cannot construct an "alternating cycle" gadget representing a graph-edge in the reduction if only a leftside-orderly set is allowed to each point.

3 Decision Problem

3.1 Algorithm for the decision problem

In this section, we present an $O((n+m)\log(n+m))$ time algorithm to solve the decision problem for a fixed scaling σ. Without loss of generality, we may assume $\sigma = 1$ in this section. We first sort the object points in nonincreasing order with respect to the x-coordinate values, and re-arrange the numbering as we noted before. We start with the following observation:

Lemma 3.1 *Let $\mathbf{L} = (L_1, L_2, \ldots, L_n)$ be a feasible placement, and let $1 \leq i \leq n$. If there is a label $L \in \mathcal{L}_i$ which is wider than L_i and intersects none of the labels $L_1, L_2, \ldots, L_{i-1}$ and the obstacles, then we can replace L_i by L to obtain another feasible placement $\mathbf{L'}$, where*

$\mathbf{L}' = (L_1, L_2, \ldots, L_{i-1}, L, L_{i+1}, \ldots, L_n)$.

From this observation, we can design a simple incremental algorithm, named DECIDE, to decide the feasibility:

Procedure DECIDE

1. For $i = 1$ to n;

1.1. If every label in $\mathcal{L}_i$ overlaps an obstacle or a label placed so far,

answer that the problem is infeasible, and exit;

1.2. Assign p_i the widest label L_i that overlaps neither any obstacle nor any label placed so far;

End For;

2. Report the placement $\mathbf{L} = (L_1, L_2, \ldots, L_n)$ as a feasible solution.

Clearly, we have:

Proposition 3.2 *Algorithm DECIDE is correct, and outputs the widest solution if the problem is feasible.*

It is easy to give a naive implementation of DECIDE with $O((n+m)n)$ time complexity. We can give a better implementation by using a standard plane sweep method, and show that it takes $O((n+m)\log(n+m))$ time. We remark that the plane sweep method is widely used in the rectangle placement and labeling problems; See for example, van Kreveld *et al.* [11].

Theorem 3.3 *The decision problem of LOFL can be solved in $O((n+m)\log(n+m))$ time.*

The decision problem is at least as difficult as the element uniqueness problem [15], and hence, if $m = O(n)$, the $O(n \log n)$ time complexity is optimal on the algebraic decision tree model.

4 Optimization problem

4.1 A precision-dependent algorithm

We consider the problem of finding the maximum feasible value of the scaling factor σ. A simple binary search algorithm on σ works; the algorithm for the decision problem can decide whether we should try a larger σ or a smaller one than the current scaling factor for the next search. If coordinate values of all points are integral and Γ is the maximum of their absolute value, then it suffices to run the algorithm given in Section 3.1 an $O(\log \Gamma)$ number of times (note that $\Gamma^2 \geq n$). Thus, we have:

Theorem 4.1 *The optimization problem of LOFL can be solved in $O((n+m)\log(n+m)\log\Gamma)$ time.*

4.2 Precision-independent algorithms

The binary search algorithm above is efficient for practical inputs for which $\log\Gamma = O(\log(n+m))$ holds. However, an efficient algorithm with time complexity independent of Γ is desirable from the theoretical point of view. We design such an $O((n+m)\log^2(n+m))$ time algorithm for the optimization problem.

4.2.1 Heterogeneous parametric searching

Meggido's parametric searching [13] is a famous method to transform a precision-dependent binary search algorithm into a precision-independent algorithm. Especially, the method is quite useful in computational geometry. (See [2, 5] and their references).

Suppose that F is a 0-1 valued *monotone* function on a parameter θ: there is a value θ_{opt} such that $F(\theta) = 1$ if $\theta \leq \theta_{opt}$ and $F(\theta) = 0$ if $\theta > \theta_{opt}$. Our aim is to compute the value θ_{opt}. Parametric searching assumes that the following two algorithms, A and D, for computing $F(\theta)$ for a given value θ are available: The algorithm D is called a *decision algorithm*, which is the fastest available algorithm to compute $F(\theta)$. Assume that D takes $O(T_D)$ time. The other algorithm A is called a *guide algorithm*. We simulate the behavior of A for $\theta = \theta_{opt}$ without knowing the value θ_{opt} in co-

operation with the decision algoirthm D, and find the value θ_{opt} in the course of the simulation. It is advantageous to use a guide algorithm that has a parallel structure, although we do not use a parallel machine in our computation. If A takes $O(t_A)$ parallel time with M processors, then we can simulate A for $\theta = \theta_{opt}$ without inputing the value θ_{opt} in $O(t_A M \log M + t_A T_D \log M)$ sequential time. Cole's acceleration method [6] can often improve the time complexity to $O(t_A M \log M + t_A T_D)$.

Let us consider our LOFL problem. We define a monotone function F as follows: $F(\sigma) = 1$ if and only if there is a feasible placement for the scaling factor σ. We can use the parametric searching paradigm regarding σ as the parameter. We use DECIDE for the decision algorithm. Unfortunately, for our problem, a guide algoirthm with $t_A = O(\log(n+m))$ and $M = O(n+m)$ seems to be difficult to design. To overcome the difficulty, we adopt a "heterogeneous" version of parametric searching. The heterogeneous parametric searching paradigm uses a "weaker" guide algorithm A that cannot compute $F(\sigma)$ itself even if σ is given as an input. Instead, A computes another function $G(\sigma)$, where the range of $G(\sigma)$ is not $\{0,1\}$ but is a much larger category. The required condition is that $G(\sigma) = G(\sigma')$ always implies $F(\sigma) = F(\sigma')$ for any σ and σ'. Intuitively, G gives a refinement of F. In particular, we will use a guide algorithm consisting of parallel sorting and point location query algorithms.

The idea of the heterogeneous parametric searching was implicitly given in Megiddo's paper [13], in which he solved a problem on the parametric minimum spanning tree of a graph by using a parallel sorting algorithm as its guide algorithm. Also, it is related to *refined parametric searching* proposed by Aggarwal *et al.*[1] for computing the minimum weight k-link path on a Monge DAG. However, to the author's knowledge, this is the first time that a heterogeneous parametric searching algorithm using a guide algorithm involving a computational geometric procedure is proposed.

4.2.2 The case without obstacles

In this subsection, we consider the case where $Q = \emptyset$. Let $\mathcal{L} = \cup_{i=1}^{n} \mathcal{L}_i$ be the set of all candidate labels, and let $S(\sigma)$ be the set of corner points of rectangles in $\mathcal{L}$ after scaled by σ and located so that the pinning points come to their corresponding object points in P. Our guide algorithm computes sorted lists $X(S(\sigma))$ and $Y(S(\sigma))$ of the points in $S(\sigma)$ with respect to the x-coordinate values and the y-coordinate values, respectively. The elements with the same ranking are tied in the sorting procedure. Using DECIDE as the decision algorithm, we compute the sorted lists $X(S(\sigma))$ and $Y(S(\sigma))$ for $\sigma = \sigma_{opt}$, where σ_{opt} is the scaling factor in the optimal solution. The following lemma is easy to prove:

Lemma 4.2 *If $X(S(\tau)) = X(S(\sigma))$ and $Y(S(\tau)) = Y(S(\sigma))$, there is a feasible solution of LOFL for the scaling factor τ if and only if there is a feasible solution for σ.*

Thus, our guide algorithm gives a refinement of the solution of the decision problem. Therefore we can find the value σ_{opt} by simulating the behavior of our guide algorithm to obtain the sorted lists $X(S(\sigma_{opt}))$ and $X(S(\sigma_{opt}))$ with help of the decision algoithm.

An $t_A = O(\log n)$ parallel time algorithm with $M = O(n)$ processors is known for the sorting problem [3]. Since the decision algorithm takes $T_D = O(n \log n)$ time, it takes $O(n \log^3 n)$ time to process the parametric search above. This can be further improved to $O(n \log^2 n)$ time by using Cole's acceleration method [6].

4.2.3 The case with obstacles

If $Q \neq \emptyset$, we need to consider overlapping of labels with obstacles. Let $V(Q)$ be the set of all vertices of polygonal obstacles in Q.

As preprocessing of our parametric searching algorithm, we prepare a point location data structure as follows: We first construct a triangulation $\mathcal{D}(Q)$ of the plane into $O(m)$ triangles so that each triangle is either contained in an obstacle or completely outside obstacles. All vertices, edges, and triangles in $\mathcal{D}(Q)$ are called *faces* of $\mathcal{D}(Q)$. Then, we prepare a point location data structure so that we can find the face of $\mathcal{D}(Q)$ containing a query point in $O(\log m)$ time. The triangulation and the point location data structure can be constructed in $O(m \log m)$ time (e.g. [15]), and we do not need to construct it in parallel since it is independent of the value of the scaling factor.

Our guide algorithm first computes the sorting lists $X(S(\sigma) \cup V(Q))$ and $Y(S(\sigma) \cup V(Q))$ of the point set $S(\sigma) \cup V(Q)$ with respect to x- and y-coordinate values, and then locates all points of $S(\sigma)$ in $\mathcal{D}(Q)$ in parallel.

A pair τ and σ of parameter values are called *equivalent* to each other if (1) $X(S(\sigma) \cup V(Q)) = X(S(\tau) \cup V(Q))$, (2) $Y(S(\sigma) \cup V(Q)) = Y(S(\tau) \cup V(Q))$ and (3) each point in $S(\tau)$ is contained in the same face of $\mathcal{D}(Q)$ as the corresponding point in $S(\sigma)$ is.

Lemma 4.3 *If σ and τ are equivalent, there is a feasible solution of LOFL for the scaling factor τ if and only if there is a feasible solution for σ.*

Hence, by simulating our guide algorithm, we can compute σ_{opt}. After constructing the point location data structure, the guide algorithm runs in $O(\log(n+m))$ time with $O(n+m)$ processors, and hence our (heterogeneous) parametric search algorithm runs in $O((n+m)\log^2(n+m))$ time, using Cole's acceleration method [6]. Thus, we have obtained the following theorem:

Theorem 4.4 *In $O((n+m)\log^2(n+m))$ time, we can find the maximum scaling factor permitting a feasible labeling of LOFL of n points in a plane with polygonal obstacles of m edges.*

5 Heuristics by using LOFL

In a practical GIS system, a map labeling problem is often given in a form which is theoretically NP-hard. Therefore, heuristics methods or hybrid methods are often effective in practice [16, 17, 18]. LOFL can be used as a powerful weapon to design heuristics combined with other methods. Suppose we have a feasible labeling with a scaling factor σ given by some method, and want to improve the factor by changing the shape of labels. Let L_i be the label for $p_i \in P$ in the labeling. In place of the single label L_i, to each $p_i \in P$, we assign an appropriate leftside-orderly set $\mathcal{L}_i$ such that $\mathcal{L}_i \ni L_i$. Thus, we have an instance of LOFL. The scaling factor in the solution of this LOFL instance is larger than or equal to σ, and is often much larger than σ. This can be considered as a "local improvement routine," which is an important tool in meta-heuristics.

6 Experimental results

For our experiment, we randomly generated 1000 instances of n integral object points in a square region of size 50000×50000 for each of $n = 20, 40, 60$, and 80. We did not locate obstacles. We tried four different labeling schemes. For each object point, we assign the following candidate labels for the respective labeling scheme: (1) A left-upper pinned rectangle with height 3σ and breadth 4σ. (2) A set of left-upper pinned rectangles of area $12\sigma^2$ whose height-breadth ratios are 12, 3, 4/3, 3/4, 1/3, and 1/12 (they correspond to factorizations of 12 to 12×1, 6×2 and 4×3). (3) A pair of rectangles with height 3σ and breadth 4σ, one of which is left-upper pinned and the other is left-lower pinned. (4) A set of rectangles consisiting of those in (2) and their reflected copies pinned at the left-lower corner.

The labeling scheme (1) can be trivially solved, and (2) is LOFL. The labeling scheme (3) can

表 1: Scaling factors of the labeling schemes

Schemes	Number of points			
	n=20	n=40	n=60	n=80
(1)	447	235	155	115
(2)	1146	643	445	333
(3)	1044	550	359	250
(4)	1477	835	598	467

be solved by using an algoirthm (2SAT) given by Forman and Wagner [7]. We solved (4) approximately by using a combination of LOFL and 2SAT. We can mathematically estimate that the expected value of the optimal scaling factor for the scheme (1) is approximately $n^{-1}\sqrt{\pi/8} \times 50000/\sqrt{12} \approx 9000n^{-1}$. Table 1 shows the average scaling factor over 1000 instances for each of (1), (2), (3), and (4) for each n.

7 Concluding remarks

We often want to place as many labels as possible for a given instance of LOFL which is infeasible for a fixed scaling factor σ. Design of efficient algoirthms or heuristics for this problem is an important future problem.

参考文献

[1] A. Aggarwal, B. Scheiber, and T. Tokuyama, Finding a minimum weight k-link path in graphs with the concave Monge property and applications, *Discrete & Computational Geometry* **12** (1994) 263–280.

[2] P. Agarwal, M. Sharir, and S. Toledo, Applications of parametric searching in geometric optimization, *Proc. 3rd ACM-SIAM Symp. on Discrete Algorithms* (1992) 72–81.

[3] M. Ajtai, J. Komlos, and E. Szemeredi, Sorting in $c\log(n)$ parallel steps, *Combinatorica*, **3** (1983), pp.1–19.

[4] H. Aonuma, H. Imai, K. Imai, and T. Tokuyama, Maximin locations of convex objects in a polygon and related dynamic Voronoi diagrams, *Proc. 6th ACM Symp. on Computational Geometry* (1990) 225–234.

[5] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir, Diameter, width, closest line pair, and parametric searching, *Proc. 8th ACM Symp. on Computational Geometry* (1992) 120–129.

[6] R. Cole, Slowing down sorting network to obtain faster sorting algorithms, *J. ACM* **34** (1987) 200–208.

[7] M. Formann and F. Wagner, A packing problem with applications to lettering of maps, *Proc. 7th ACM Symp. on Computational Geometry* (1991) 281–290.

[8] C. Iturriaga and A. Lubiw, Elastic labels around the perimeter of a map, *Proc. WADS'99* (1999)

[9] K. Kakoulis and I. Tollis, A unified approach to labeling graphical features, *Proc. 14th ACM Symp. on Computational Geometry* (1998) 347–356.

[10] K. Kato, Studies on the Geometric Location Problems, L1 Approximation and Character Placing, Master Thesis, Kyushu University (February 1989).

[11] M. van Kreveld, T. Strijk, and A. Wolff, Point set labeling with sliding labels, *Proc. 14th ACM Symp. on Computational Geometry* (1998) 337–346.

[12] M. Halldórsson, private communication (1999).

[13] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *J. ACM* **30** (1983) 852–865.

[14] I. Shamos and F. Preparata, *Computational Geometry – An Introduction*, Springer Verlag, 1985.

[15] F. Wagner and A. Wolff, Map labeling heuristics: provably good and practically useful, *Proc. 11th ACM Symp. on Computational Geometry* (1995) 109-118.

[16] F. Wagner and A. Wolff, A combinatorial framework for map labeling, *Proc. Graph Drawing '98* (1998) 13–15.

[17] M. Yamamoto, G. Camara, L. Lorena, Tabu Search Heuristics for Point-Feature Cartographical Label Placement, *GeoInformatica* (2000) (also see http://www.lac.inpe.br/~lorena/missae/index.html)