Approximation Algorithms for Aligning Points

Sergio Cabello

Marc van Kreveld

institute of information and computing sciences, utrecht university

technical report UU-CS-2003-003

www.cs.uu.nl

Approximation Algorithms for Aligning Points *

Sergio Cabello Marc van Kreveld

Institute of Information and Computing Sciences
Utrecht University
The Netherlands
{sergio,marc}@cs.uu.nl

Abstract

We study the problem of aligning as many points as possible horizontally, vertically, or diagonally, when each point is allowed to be placed anywhere in its own, given region. Different shapes of placement regions and different sets of alignment orientations are considered. More generally, we assume that a graph is given on the points, and only the alignments of points that are connected in the graph count. We show that for planar graphs the problem is NP-hard, and we provide an inapproximability result for general graphs. For the case of trees and planar graphs, we give approximation algorithms whose performance depends on the shape of the given regions and the set of orientations. When the orientations are the ones given by the axes and the regions are axis-parallel rectangles, we obtain a polynomial time approximation scheme.

1 Introduction

Placement problems for geometric objects have appeared in various forms in computational geometry. A particular position is sought that optimizes some measure and/or satisfies certain criteria. Facility location is an obvious example, where a placement of a point is desired that minimizes, for instance, the sum of distances to all other points in a given set. When the maximum distance is minimized, the problem is the well-known smallest enclosing disk problem [6, 18]. In motion planning, the positioning of a robot in the free configuration space requires the placement of a polygon amidst other polygons without intersection [17].

There are also geometric placement problems where many objects must be placed. Some examples are the placement of shapes for clothing manufacturing [12, 5] such that the amount of fabric lost is minimized, label placement on maps [19], guard placement [16], and graph drawing [7]. Some of these problems are related to packing. Often, placement problems for multiple objects are computationally demanding, because the problems have many degrees of freedom in the solution space. Especially when some measure must be optimized, such problems are generally NP-hard, and therefore polynomial-time algorithms are not known to exist.

This paper studies another placement problem for multiple objects, motivated from cartography. In the design of schematic networks, like subway maps, a strongly simplified depiction of a transportation system should be computed. The connection between two major locations or junctions is shown in a stylized manner where the exact geometry of the connection is unimportant. Figure 1, left, shows an example.

Instead of faithful geometry, a connection is usually shown using only a small number of segments which all have one of four orientations: horizontal, vertical, or one of the two diagonals. The automated construction of schematic maps has been studied in several papers [1, 3, 4, 8, 15]. Some of the proposed methods leave the positions of the junctions untouched and concentrate

*Supported by the Dr.ir. Cornelis Lely Stichting.

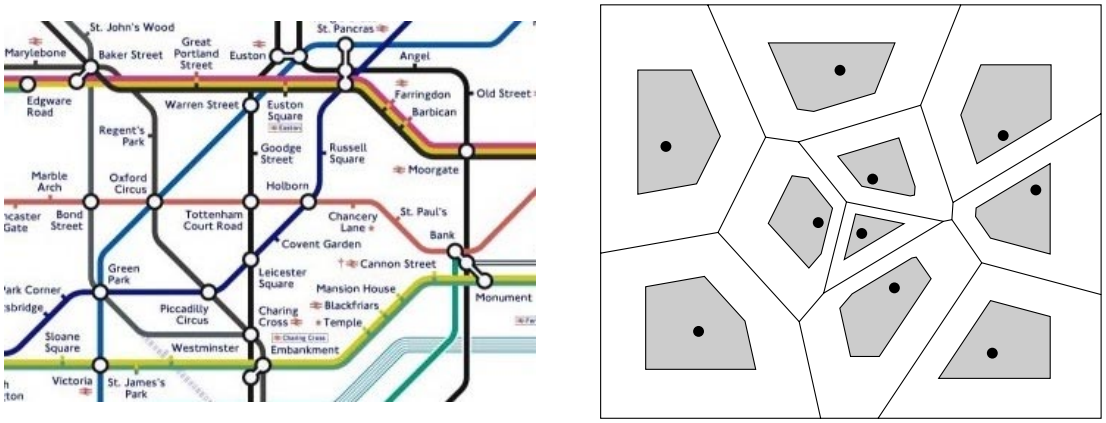

Figure 1: Left: Detail of the underground map of London. Right: Voronoi diagram of points and scaled Voronoi cells.

on the schematization of the connections only. Other suggested methods consist of iterative approaches where the connections should converge to the major orientations, while displacing the junctions. Methods of the latter type don't bound the maximum displacement, and convergence is not guaranteed.

This paper presents a combinatorial method to displace the important locations or junctions of a schematic network. We abstract the problem as follows: Let a set P of n points in the plane be given, and for each point $p_i \in P$ some region S_i around it. Furthermore, a graph is given of which the nodes correspond one-to-one with the points of P (and the regions). Find for each point p_i a position in its region such that the number of alignments with other points of P is maximized. Here alignment is for a given, constant number of orientations, and alignment only counts (is optimized) for two points whose nodes are connected in the graph.

The motivation for abstracting the alignment problem for schematic networks this way is as follows. The precise positions of the junctions in the schematic network is not important, but the positions must be approximately preserved. Hence the introduction of a region around each point. Alignment on schematic networks usually implies horizontal or vertical alignment, or also diagonal (45 or 135 degrees) alignment. This is abstracted to alignment with respect to a constant number of given orientations. Of course, alignment is important only for two points that have a connection. This is modelled by the graph, which generally is a planar or almost planar graph.

For the type of regions around each point, there are various natural choices. A fixed, maximum allowed displacement gives rise to a fixed radius disk around each point. Because the preservation of the approximate East-West positioning and North-South positioning is more important than for any other direction, we could instead choose squares or rectangles. Since the relative positioning to points in the neighborhood is important, one could also choose to allow each point to be placed anywhere in its Voronoi cell, or in a scaled-down copy of it (Figure 1, right). This allows points further away from other points to be displaced more than points in a cluster, a behavior that is desirable. Hence, in our alignment problems we will consider circular, rectangular, and convex polygonal regions in the problem statement. We do not deal with the actual choice of the regions, or which regions are preferable, but we assume that the regions are already given.

An interesting aspect of the problem is that it contains both geometry and graph aspects. We will combine ideas from both fields in this paper. In next section we formalize the problem and show that if we are able to approximate the optimal solution when the graph is a tree, then we also obtain an approximation for planar graphs. In Section 3 we show that a rather simple version of the problem where we only care about vertical alignment is NP-hard. We also give an inapproximability result for general graphs provided that P$\neq$NP.

In Sections 4, 5, and 6, we give approximation algorithms for different cases of the problem.

orientations	graph, region	approximation ratio	time	reference
1	tree	1	$O(n^2)$	Theorem 4
	planar graph	$\frac{1}{3}$	$O(n^2)$	Corollary 3
		$\frac{k-1}{k}$	$O(k(2n)^{3k+1})$	Theorem 7
1	tree, convex	$\frac{k}{k+1}$	$O(k9^k n^2)$	Theorem 3
	planar graph, convex	$\frac{k}{3(k+1)}$	$O(k9^k n^2)$	Corollary 2
	tree, rectangles	1	$O(n^3)$	Theorem 5
	planar graph, rectangles	$\frac{1}{3}$	$O(n^3)$	Corollary 5
		$\frac{k-1}{k}$	$O(k(2n)^{6k+1})$	Theorem 8
	planar graph, disjoint rectangles	$\frac{1}{3}$	$O(n^2)$	Corollary 4
		$\frac{k-1}{k}$	$O(k(2n)^{3k+1})$	Corollary 9
$h > 2$	planar graph, convex	$\frac{1}{3h}$	$O(n^2)$	Corollary 6
		$\frac{2k}{3h(k+1)}$	$O(k9^k n^2)$	Corollary 7
		$\frac{k}{3(k+1)}$	$n^{O(2^k)}$	Theorem 6
		$\frac{k}{h(k+1)}$	$O(k(2n)^{3k+1})$	Corollary 8

Table 1: Approximation algorithms in this paper.

Both the approximation factors and the time bounds depend on the properties of the regions and the set of orientations; the results are summarized in Table 1. More specifically, in Section 4 we give a polynomial time approximation scheme (PTAS) when the graph is a tree. In Section 5 we use the same approach to get several approximation algorithms for planar graphs. For the case of rectangular regions and the horizontal and vertical orientations only, we give a polynomial time approximation scheme as well. This is based on the results from Baker [2] and is explained in Section 6. We finish with the conclusions and open problems.

2 Preliminaries

2.1 Formulation of the problem

In this section we formalize the problem of alignment. Recall that a graph $G = (\mathcal{S}, E)$ consists of a set of nodes $\mathcal{S}$ and a set of edges $E \subset \binom{\mathcal{S}}{2}$. In particular, we consider graphs where each node is a convex region. Given a fixed set of orientations O, we define a function χ_O that assigns to pairs of regions the value 1 if there is a line with orientation in O that intersects both regions, and 0 otherwise. In particular, for two points p, q, we have $\chi_O(p, q) := \chi_O(\{p\}, \{q\}) = 1$ if the line through p and q has its orientation in O, and 0 otherwise. For the application to cartography, the orientations will typically be axis-parallel ($|O| = 2$) or also including diagonal lines (with slope 1 or -1, so $|O| = 4$).

The problem can be stated as follows: given a set of n convex regions, $\mathcal{S} = \{S_0, \ldots, S_{n-1}\}$, a graph $G = (\mathcal{S}, E)$ on those regions, and a set of orientations O, place n points $p_0, \ldots, p_{n-1}$ with $p_i \in S_i$ to maximize the function

$$\sum_{\{S_i, S_j\} \in E} \chi_O(p_i, p_j).$$

We denote the maximum value by $M_O(G)$, or simply $M(G)$, as we consider the given orientations O to be fixed. A $\frac{1}{r}$-approximation of $M_O(G)$, where $r \geq 1$, is a collection of n points $p_0, \ldots, p_{n-1}$ with $p_i \in S_i$ such that

$$\sum_{\{S_i, S_j\} \in E} \chi_O(p_i, p_j) \geq \frac{1}{r} M_O(G)$$

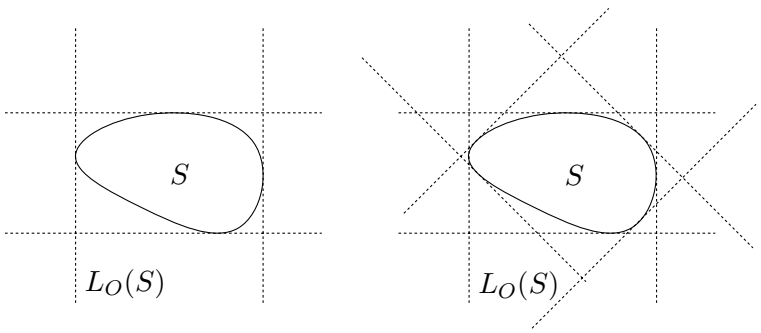

Figure 2: Left: $L_O(S)$ for a region S when O is axis-parallel. Right: $L_O(S)$ when O is axis-parallel and diagonal

For our application to cartography, we usually assume G to be a planar graph. Nevertheless, we also will consider in some extension non-planar graphs. Typical regions S_i that we consider are scaled Voronoi cells, rectangles, and circles. However, it turns out that we only need to distinguish the case of axis-parallel rectangles and any other convex region. Regions can overlap or not, which leads to slightly different results. When the regions overlap, the placement of two points can coincide, and in this case we also assume that they are aligned.

We remark that possibly, the computed placement does not give a planar straight-line embedding. In fact we are not assuming that an embedding is given initially. If this would be the case, the new embedding may be non-equivalent to the original one.

For a region S, we define $L_O(S)$ to be the set of lines tangent to S that have orientation in O (see Figure 2). In the algorithm to be described, we will subdivide region S into cells $C^1, \ldots, C^t$. We will also use the notation $L_O(C^j)$ for the lines with orientation in O that are tangent to the cell C^j. For a set L of lines, we will use $\mathcal{A}(L)$ for the arrangement in the plane induced by L (see [6] for the concept).

2.2 Decomposing the original graph

It appears to be difficult to develop a general technique that gives a good approximation algorithm for any graph G, any shape of region, and any set of alignment orientations. But if G is a tree, we will present a general approach in Section 4 that gives several different polynomial time approximation results, depending on the shape of the regions and the number of alignment orientations. Furthermore, it is known that a planar graph G can be decomposed into three trees (or forests), such that every edge of G appears in exactly one tree (or forest) [10]. Such partition can be found in $O(n \log n)$ time, and is the main ingredient for the following result.

Lemma 1 *Given $r \geq 1$, if for any tree $\mathcal{T}$ we can compute a $\frac{1}{r}$-approximation of $M_O(\mathcal{T})$ in $O(T(n))$ time, then we can compute a $\frac{1}{3r}$-approximation of $M_O(G)$ for any planar graph G in $O(T(n) + n \log n)$ time.*

Proof: Given a planar graph G, we decompose it into three edge disjoint forests $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$. Let A_i be a $\frac{1}{r}$-approximation of $M_O(\mathcal{F}_i)$. The value $A := \max\{A_1, A_2, A_3\}$ can be computed in $O(n \log n + 3T(n) + 1) = O(n \log n + T(n))$ time, and it is a $\frac{1}{3r}$-approximation of $M_O(G)$. Indeed, consider the placement $p_0, \ldots, p_{n-1}$ that achieves $M_O(G)$. Then

$$M_O(G) = \sum_{\{S_i,S_j\} \in E} \chi_O(p_i, p_j) = \sum_{k=1}^{3} \Big(\sum_{\{S_i,S_j\} \in \mathcal{F}_k} \chi_O(p_i, p_j) \Big) \leq \sum_{k=1}^{3} M_O(\mathcal{F}_k),$$

and because A_i is a $\frac{1}{r}$-approximation of $M_O(\mathcal{F}_i)$ we get

$$M_O(G) \leq \sum_{k=1}^{3} M_O(\mathcal{F}_k) \leq \sum_{k=1}^{3} rA_i \leq 3rA.$$

□

So, basically, when we approximate the original problem for the special case of trees we also obtain an approximation for a planar graph. The same approach also works for general graphs. In [9] it is shown how to get an edge-disjoint partition of a graph in $O(kn^{\frac{3}{2}}\sqrt{n + k \log n})$ time, where k is the number of forests needed. However, since k can be $\Omega(n)$, the approximation ratio for a general graph would be $O(\frac{1}{n})$ times the one for trees, which is not really interesting.

3 Hardness of the problem

We show the hardness of a rather simple version of the aligning problem: the regions are horizontal segments and we want to maximize the number of vertical alignments (so $|O| = 1$). The reduction is from E3-SAT (Exact3-SATifiability), and implies an inapproximability result for non-planar graphs. An E3-SAT instance is a formula of t Boolean variables $x_1, \ldots, x_t$ given by m conjunctive clauses $C_1, \ldots, C_m$, where each clause contains exactly 3 literals (a variable or its negation). MAX-E3-SAT is the associated optimization problem: given an E3-SAT instance, find an assignment to the variables that maximizes the number of satisfied clauses.

Theorem 1 *Let O be the vertical orientation, let $\mathcal{S} = \{S_0, \ldots S_{n-1}\}$ be a set of horizontal segments and let $G = (\mathcal{S}, E)$ be a graph. For any $\epsilon > 0$, it is NP-hard to place points $p_0, \ldots, p_{n-1}$ with $p_i \in S_i$ that yield a $(\frac{15}{16} + \epsilon)$-approximation of $M_O(G)$*

Proof: Given a E3-SAT instance ϕ with t variables $x_1, \ldots, x_t$ and m clauses $C_1, \ldots, C_m$, we construct an aligning problem $\mathcal{P}_\phi$ as follows (see Figure 3):

1. take $\mathcal{S} := \emptyset, E := \emptyset$;
2. for each Boolean variable x_i, add the horizontal interval $I_i := [i - \frac{1}{3}, i + \frac{1}{3}]$ to $\mathcal{S}$;
3. for each clause C_j, add the horizontal interval $J_j := [\frac{2}{3}, t + \frac{1}{3}]$ to S;
4. for each occurrence of x_i in C_j, add the horizontal interval $I_{i,j} := [i - \frac{1}{3}, i - \frac{1}{3}] = \{i - \frac{1}{3}\}$ to $\mathcal{S}$, and the edges $\{J_j, I_i\}$ and $\{J_j, I_{i,j}\}$ to E;
5. for each occurrence of the negation of x_i in C_j, add the horizontal interval $I_{i,j} := [i + \frac{1}{3}, i + \frac{1}{3}] = \{i + \frac{1}{3}\}$ to $\mathcal{S}$ and the edges $\{J_j, I_i\}$ and $\{J_j, I_{i,j}\}$ to E.

When considering a placement in this aligning problem $\mathcal{P}_\phi$, we can assume that all points have the x-coordinate in the set $C = \{1 - \frac{1}{3}, 1 + \frac{1}{3}, \ldots, t - \frac{1}{3}, t + \frac{1}{3}\}$. If a point has a different x-coordinate, we displace it to the largest x-coordinate value in C that is smaller than the actual value. By doing this, we always keep or increase the number of alignments: two points that were vertically aligned keep being vertically aligned because either they were not displaced or they both have been displaced to the same x-coordinate. With this assumption, we have a bijection between the Boolean assignments of the variables $x_1, \ldots, x_t$ and the placements of the points $p_1, \ldots, p_t$ with $p_i \in I_i$: x_i is true if and only if $p_i \in I_i$ is placed at $i - \frac{1}{3}$, and false if and only if $p_i \in I_i$ is placed at $i + \frac{1}{3}$.

Consider an assignment of the Boolean variables $x_1, \ldots, x_t$ and the corresponding placement of points in the regions $I_1, \ldots, I_t$. The key observation is that a clause C_j is satisfied in the assignment if and only if we can place a point in the region J_j that provides two alignments. When C_j is not satisfied, the placement of a point in the region J_j provides exactly one alignment. Therefore, we can satisfy s clauses in ϕ if and only if we can align $m + s$ pairs of points in the

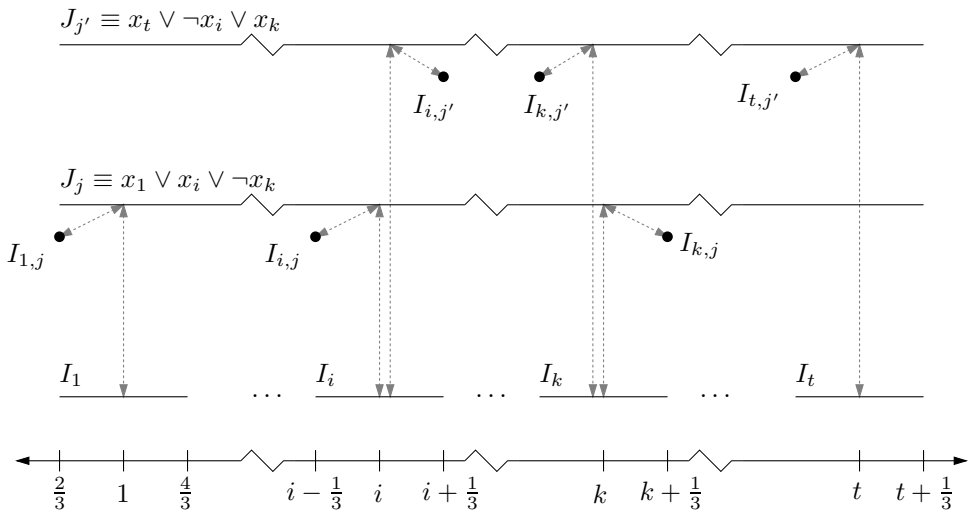

Figure 3: Reduction from E3-SAT to an aligning problem. The variables $x_1, \ldots, x_t$ are represented by the segments $I_1, \ldots, I_t$, and each clause C_j is represented by the segment J_j plus three segments $I_{i,j}$ that depend on its literals.

corresponding problem $\mathcal{P}_\phi$. In particular, for a satisfiable instance ϕ, the optimum number of alignments is $2m$.

If we have a polynomial time $(\frac{15}{16} + \epsilon)$-approximation algorithm for the aligning problem, and we use it for $\mathcal{P}_\phi$, where ϕ is a satisfiable E3-SAT instance, we would get at least

$$(\frac{15}{16} + \epsilon)2m = \frac{30m}{16} + 2\epsilon m = m + (\frac{7}{8} + 2\epsilon)m$$

alignments. But then we would have a polynomial time $(\frac{7}{8} + 2\epsilon)$-approximation algorithm for MAX-E3-SAT on satisfiable instances, which is NP-hard by Theorem 6.5 of [11]. □

Corollary 1 *If the graph G is planar, it is NP-hard to compute $M_O(G)$.*

Proof: Consider a planar 3-SAT instance ϕ (see [14] for the definition) and apply the reduction used in the proof of the previous theorem to get an aligning problem $\mathcal{P}_\phi$. We claim that the graph G_ϕ of the problem $\mathcal{P}_\phi$ is planar. Observe that the nodes of the type $I_{i,j}$ have degree one, and therefore we can remove them without affecting the planarity or non-planarity of the graph. The remaining graph is

$$(\{I_1, \ldots, I_t, J_1, \ldots, J_m\}, \{\{I_i, J_j\} |\, x_i \text{ appears in } C_j\}),$$

and is isomorphic to

$$(\{x_1, \ldots, x_t, C_1, \ldots, C_m\}, \{\{x_i, C_j\} |\, x_i \text{ appears in } C_j\}),$$

which has to be planar by definition of planar 3-SAT instances.

As discussed in the previous proof, $M_O(G_\phi) = 2m$ if and only if ϕ is satisfiable. Therefore, if for any planar graph we can compute $M(G)$, we can decide the satisfiability of planar 3-SAT instances, which is NP-hard [14]. □

It is natural to wonder if we can construct aligning problems where it is NP-complete to decide if all edges can be aligned or not. We can show that the answer is negative if we restrict ourselves to only one orientation.

Theorem 2 *Let O be the vertical orientation, let $\mathcal{S} = \{S_0, \ldots S_{n-1}\}$ be a set convex regions and let $G = (\mathcal{S}, E)$ be a connected graph. Then, $M_O(G) = |E|$ if and only if there is a vertical line that intersects all regions.*

Proof: We can assume that the regions are horizontal segments, otherwise, we project each regions onto a horizontal line and we get an equivalent problem.

If all the intervals in G are intersected by a vertical line l, then we can place the point $p_i \in S_i$ at $l \cap S_i$. It is clear that we get $M_O(G) = |E|$ because all points are vertically aligned.

For the other implication, consider a placement $p_0, \ldots, p_{n-1}$, with $p_i \in S_i$, that achieves $M_O(G) = |E|$ alignments. We claim that the vertical line through p_0 goes also through all p_i. To see this, fix any S_i, and assume without loss of generality that $S_0, S_1, \ldots, S_i$ is a path in G from S_0 to S_i (it always exists because G is connected). Then, point p_0 has to be vertically aligned with point p_1, and p_1 has to be vertically aligned with p_2, and so on until p_i. Because being vertically aligned is a transitive relation, p_0 has to be vertically aligned with p_i, and both are contained in the same vertical line. □

4 The basic approach for trees

We explain the algorithm for alignment for the specific case of convex regions and two aligning orientations. In next section we will analyze what results are obtained when we apply the same technique to other versions of the alignment problem, with different region shapes and different alignment orientations.

Let $\mathcal{S} = \{S_0, \ldots, S_{n-1}\}$ be a set of n convex regions, and let $\mathcal{T} = (\mathcal{S}, E)$ be a tree. We choose any node, S_0, to be the root of $\mathcal{T}$. Let b_i be the complexity of the boundary of region S_i. If for an arbitrary node S_i, we remove from $\mathcal{T}$ the edge connecting S_i with its parent node, we get two subtrees. We will use $\mathcal{T}_i$ to denote the subtree containing the node S_i. We assume that nodes $S_1, \ldots, S_d$ are the neighbors of node S_0, so d is the degree of S_0. In particular, when we remove S_0 from $\mathcal{T}$, we get the subtrees $\mathcal{T}_1, \ldots, \mathcal{T}_d$ (see Figure 4, left). We use $\mathcal{T}(p_i)$ to denote the graph $\mathcal{T}$ after replacing the node S_i by p_i, that is, the point p_i is the placement chosen for the region S_i (see Figure 4, right). Fixing a point p_0 in the region S_0 makes the subproblems that appear in the subtrees $\mathcal{T}_1, \ldots, \mathcal{T}_d$ independent, and therefore we get the following recurrence:

$$M(\mathcal{T}(p_0)) = \sum_{i=1}^{d} \max_{p_i \in S_i} \{\chi_O(p_0, p_i) + M(\mathcal{T}_i(p_i))\}.$$

The overall idea is to subdivide (or partition) region S_0 into cells such that any placement within a cell will give exactly the same solution. This will be done in a recursive way: to construct the subdivision in S_0 we will use subdivisions of $S_1, \ldots, S_d$ with that same property, but only for the corresponding subtree: each placement in a cell of S_i gives the same number of alignments in $\mathcal{T}_i$.

Definition 1 *A convex cell $C \subseteq S_i$ is* $\mathcal{T}$-stable *if and only if*

$$M(\mathcal{T}(p_i)) = M(\mathcal{T}(p_i')) \qquad \forall p_i, p_i' \in C.$$

We use $M(\mathcal{T}(C))$ to denote this invariant value.

It is clear that if S_i is a leaf of $\mathcal{T}$, then S_i already is a $\mathcal{T}_i$-stable cell. This gives the basis for a recursive formulation on how to make the subdivision of S_0. Let $C_i^1, \ldots, C_i^{t_i}$ be a subdivision of S_i into $\mathcal{T}_i$-stable cells. Let L_0 be the set of all lines with orientation in O that are tangent to some cell C_i^j, where $i = 1 \ldots d$ and $j = 1 \ldots t_i$ (see the example in Figure 5). In other words, we have

$$L_0 = \bigcup_{i=1}^{d} \bigcup_{j=1}^{t_i} L_O(C_i^j).$$

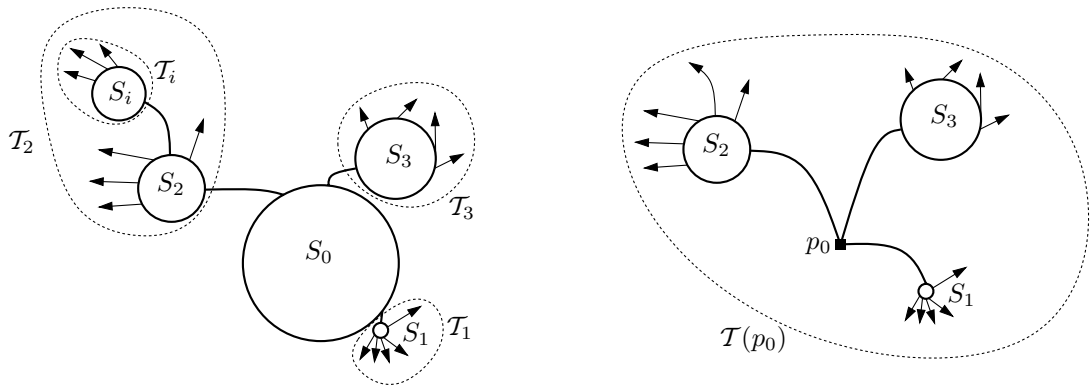

Figure 4: Left: Neighbors of S_0 and the corresponding subtrees. Right: When we substitute S_0 by $\{p_0\}$ we get $\mathcal{T}(p_0)$.

We can subdivide S_0 using all lines in L_0 to make an arrangement $\mathcal{A}(L_0)$ inside S_0.

Lemma 2 *Any cell in $\mathcal{A}(L_0) \cap S_0$ is $\mathcal{T}$-stable.*

Proof: Consider any cell C in $\mathcal{A}(L_0) \cap S_0$, and two points $p_0, p_0' \in C$. We want to show that $M(\mathcal{T}(p_0)) = M(\mathcal{T}(p_0'))$. Let the points $p_1, \ldots, p_d$ with $p_i \in S_i$ be placed to attain the value $M(\mathcal{T}(p_0))$, that is,

$$M(\mathcal{T}(p_0)) = \sum_{i=1}^{d} \{\chi_O(p_0, p_i) + M(\mathcal{T}_i(p_i))\}.$$

Let $C_i^{j_i} \subset S_i$ be the $\mathcal{T}_i$-stable cell in which p_i lies. If we have points $p_1', \ldots, p_d'$ with $p_i' \in C_i^{j_i}$ such that $\chi_O(p_0, p_i) \leq \chi_O(p_0', p_i')$, then

$$M(\mathcal{T}(p_0)) = \sum_{i=1}^{d} \{\chi_O(p_0, p_i) + M(\mathcal{T}_i(p_i))\} \leq \sum_{i=1}^{d} \{\chi_O(p_0', p_i') + M(\mathcal{T}_i(p_i))\}$$

and because p_i and p_i' are in the same $\mathcal{T}_i$-stable cell $C_i^{j_i}$

$$M(\mathcal{T}(p_0)) \leq \sum_{i=1}^{d} \{\chi_O(p_0', p_i') + M(\mathcal{T}_i(p_i))\} = \sum_{i=1}^{d} \{\chi_O(p_0', p_i') + M(\mathcal{T}_i(p_i'))\} \leq M(\mathcal{T}(p_0')).$$

But by symmetry we also have $M(\mathcal{T}(p_0')) \leq M(\mathcal{T}(p_0))$ and therefore $M(\mathcal{T}(p_0)) = M(\mathcal{T}(p_0'))$.

The points $p_1', \ldots, p_d'$ that we need can be found as follows. If $\chi_O(p_0, p_i) = 0$, take $p_i' := p_i$, and the properties hold. If $\chi_O(p_0, p_i) = 1$, let $o \in O$ be the orientation of the line $\overline{p_0 p_i}$, and let l_i be the line through p_0' with orientation o. Because p_0 and p_0' are in the same cell C of $\mathcal{A}(L_0) \cap S_0$, the line l_i lies between the two tangents to $C_i^{j_i}$ with orientation o. Therefore, the intersection $C_i^{j_i} \cap l_i$ is nonempty, and any $p_i' \in C_i^{j_i} \cap l_i$ has the desired properties. □

When we have subdivided S_0 into $\mathcal{T}$-stable cells $C_0^1, \ldots C_0^{t_0}$, we can compute the maximum value $M(\mathcal{T}) = \max_{j \in \{1,\ldots,t_0\}} \{M(\mathcal{T}(C_0^j))\}$. Thus, we need to be able to compute, for a $\mathcal{T}$-stable cell C, the actual number of alignments $M(\mathcal{T}(C))$: we place an arbitrary point $p_0 \in C$ and then we have

$$M(\mathcal{T}(C)) = M(\mathcal{T}(p_0)) = \sum_{i=1}^{d} \max_{j \in \{1 \ldots t_i\}} \{\chi_O(p_0, C_i^j)) + M(\mathcal{T}_i(C_i^j))\}.$$

There are two important issues to address: how many cells does the subdivision of S_0 have if we recursively use Lemma 2, and how much time does it take to compute the value $M(\mathcal{T}(C))$

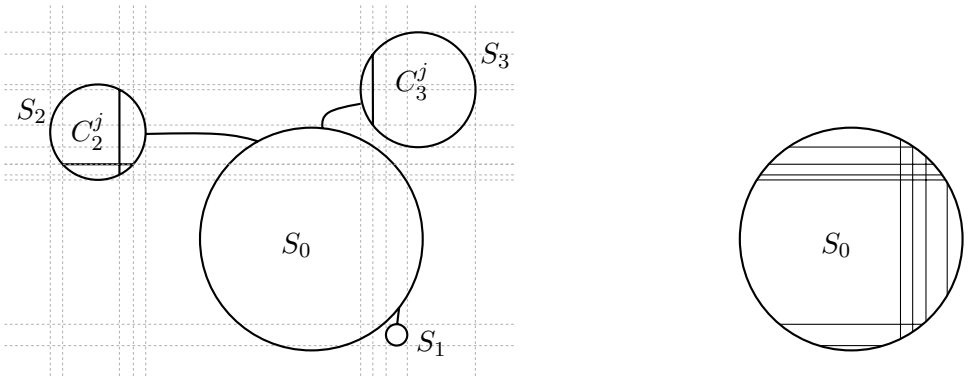

Figure 5: In this example S_1, S_2, S_3 are adjacent to S_0 in $\mathcal{T}$. The tangents to the cells C_i^j induce a subdivision in S_0.

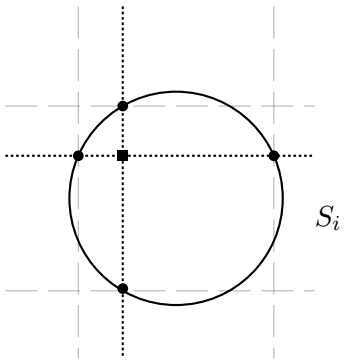

Figure 6: Analysis of the different tangents that can be produced. The vertex represented by a square is internal to S_i, and doesn't produce more tangent lines than we had. The other vertices come from an intersection of a line in L_i with the boundary of S_i, and it contributes to L_0 with at most one new line. Furthermore, the region has $2|O| = 4$ tangents that are new lines.

for each cell C of the subdivision. We will bound the time spent at node S_0 assuming we have already processed its children $S_1, \ldots, S_d$.

Lemma 3 *If the tree $\mathcal{T}$ has height k, then we can subdivide S_0 into $O(9^k n^2)$ $\mathcal{T}$-stable cells in $O(9^k n^2 + b_0)$ time, where b_0 is the complexity of S_0.*

Proof: We compute the set of lines L_0 that has been used in the previous lemma, and then we compute $\mathcal{A}(L_0) \cap S_0$. We start by bounding the number of lines in L_0. Let L_i be the set of lines that are used in the recursive process to subdivide the region S_i into $\mathcal{T}_i$-stable cells. Any line in L_0 is tangent to some vertex of a cell C_i^j, where $i \in \{1, \ldots, d\}$. Three cases arise (see Figure 6):

- The vertex is interior to S_i, that is, it was determined by the intersection of two of the lines in L_i. Because we only have two orientations, those tangents are already present in L_i.
- The vertex is on the intersection of the boundary of S_i and a line in L_i. The region S_i is convex, so any line in L_i intersects the boundary at most twice. Therefore each line can produce at most two new lines in L_0.
- The vertex is on the boundary, but it does not lie on any line of L_i. In this case each region S_i can produce at most $2|O| = 4$ new lines.

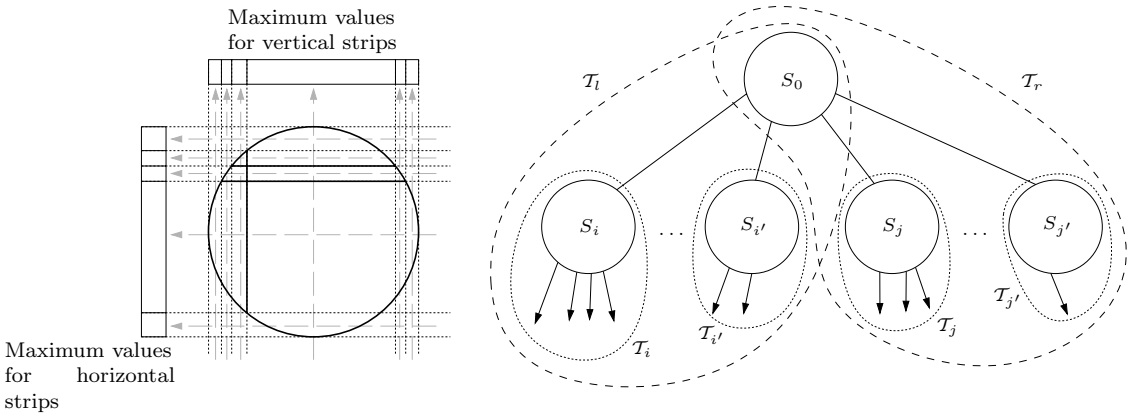

Figure 7: The proof of Lemma 4. Left: If S_0 only has one child, we store in each strip of S_1 the maximum values. Right: If S_0 has more than one child we use divide and conquer.

Therefore, we have the recursive relation $|L_0| \leq \sum_{i=1}^{d}(3|L_i|+4)$, and we will show that $|L_0| \leq 3^{k+1}(n-1) = O(3^k n)$ by induction on the height k of the tree. Indeed, if the tree has height 0, then it consists of only one node and it is trivially true because no tangent line has been used. For the general case, observe that the set L_i of lines has been constructed recursively from the tree $\mathcal{T}_i$, which is rooted at node S_i and has height $k-1$. Therefore, if $|\mathcal{T}_i|$ denotes the number of nodes in $\mathcal{T}_i$, we have

$$|L_0| \leq \sum_{i=1}^{d}(3|L_i|+4) \leq 4d + 3\sum_{i=1}^{d} 3^k(|\mathcal{T}_i|-1) = 4d + 3^{k+1}\sum_{i=1}^{d}(|\mathcal{T}_i|-1))$$

and because $\sum_{i=1}^{d}|\mathcal{T}_i| = n-1$ and $k \geq 1$ we get

$$|L_0| \leq 4d + 3^{k+1}(n-d-1) = 3^{k+1}(n-1) + d(4-3^{k+1}) \leq 3^{k+1}(n-1).$$

This finishes the inductive proof that shows $|L_0| = O(3^k n)$.

To construct L_0, we need to find, for each child S_i of S_0, the intersections of L_i with the boundary of S_i. But this has been done already when $\mathcal{A}(L_i) \cap S_i$ was computed, and therefore takes time linear in the number of lines generated. Once we have L_0, we compute $\mathcal{A}(L_0)$ and walk through the boundary of S_0 to compute $\mathcal{A}(L_0) \cap S_0$, the portion of the arrangement $\mathcal{A}(L_0)$ inside S_0. We can bound the time spent in this part by $O(b_0)$ (the complexity of the boundary of S_0) plus the complexity of the arrangement, which is $O(b_0 + (3^k n)^2) = O(9^k n^2 + b_0)$ in total. □

Once we have computed all $\mathcal{T}$-stable cells $C_0^1, \ldots, C_0^{t_0}$, we can compute the values $M(\mathcal{T}(C_0^j))$. Assuming that each neighbor S_i of S_0 has been subdivided into $C_i^1, \ldots, C_i^{t_i}$ $\mathcal{T}_i$-stable cells, and that the corresponding values $M(\mathcal{T}_i(C_i^1)), \ldots, M(\mathcal{T}_i(C_i^{t_i}))$ have been computed, we will show how to compute the values of the cells in the subdivision of S_0. Observe that if we would compute for each cell C_0^j the value $M(\mathcal{T}(C_0^j))$ by examining the children, then we would spend $\Omega(d)$ time per cell, and so it would take $\Omega(9^k n^2 d)$ time. Because d can be $\Omega(n)$, this gives $\Omega(9^k n^3)$ time in the worst case. We can do better than this using a divide and conquer approach on the children of S_0.

Lemma 4 *We can compute the values $M(\mathcal{T}(C_0^1)), \ldots, M(\mathcal{T}(C_0^{t_0}))$ in $O(9^k n^2 + d b_0)$ time.*

Proof: Let $T(n,d)$ be the time needed when $\mathcal{T}$ has n nodes and S_0 has d children in $\mathcal{T}$. There are two cases depending on the value of d:

- If $d = 1$, then S_0 has only one child, S_1. Let $C_1^1, \dots C_1^{t_1}$ be the subdivision on S_1 into $\mathcal{T}_1$-stable cells. Then, for every strip of the subdivision of S_1 with orientation in O, we compute the maximum value $M(\mathcal{T}(C_1^j))$ over all cells C_1^j in that strip and store it in one of two arrays, one for each orientation (see Figure 7, left). We also store the maximum value over all cells $M_1 := \max\{M(C_1^1), \dots, M(C_1^{t_i})\}$. This can be done in $O(9^{k-1}n^2)$ time.

  We already had $\mathcal{A}(L_0) \cap S_0$, and now, for each cell $C_0^j \in \mathcal{A}(L_0) \cap S_0$, we take a point $p_0 \in C_0^j$:

$$M(\mathcal{T}(C_0^j)) = M(\mathcal{T}(p_0)) = \max_{j \in \{1 \dots t_1\}} \{\chi_O(p_0, C_1^j)) + M(\mathcal{T}_1(C_1^j))\} =$$

$$= \max_{o \in O} \Big\{ M_1, 1 + \max_{C_1^j,\, \chi_{\{o\}}(p_0, C_1^j)=1} \{M(\mathcal{T}_1(C_1^j))\} \Big\}.$$

  But the value $\max_{C_1^j,\, \chi_{\{o\}}(p_0,C_1^j)=1}\{M(\mathcal{T}_1(C_1^j))\}$ corresponds to an entry in the array corresponding to the orientation o, so it takes constant time to compute $M(\mathcal{T}(C_0^j))$. Because $\mathcal{A}(L_0) \cap S_0$ has $O(9^k n^2)$ cells, we conclude that $T(n, 1) = O(9^k n^2)$.

- If $d > 1$, then S_0 has more than one child. In this case, we split its children into two sets $\mathcal{S}_l$ and $\mathcal{S}_r := \{S_1, \dots, S_d\} \setminus \mathcal{S}_l$, and consider the subtrees $\mathcal{T}_l$ and $\mathcal{T}_r$, where $\mathcal{T}_l$ is the connected component of $\mathcal{T} \setminus \mathcal{S}_r$ that contains S_0 and $\mathcal{T}_r$ is the connected component of $\mathcal{T} \setminus \mathcal{S}_l$ that contains S_0 (see Figure 7, right). Let $L_l \subset L_0$ be the set of lines that have been produced by nodes $S_i \in \mathcal{S}_l$ in Lemma 3, and let $L_r \subset L_0$ be the set of lines that have been produced by nodes $S_j \in \mathcal{S}_r$ in Lemma 3, thus we have $L_0 = L_l \cup L_r$. By Lemma 2, any cell $C_l \in \mathcal{A}(L_l) \cap S_0$ is $\mathcal{T}_l$-stable and any cell $C_r \in \mathcal{A}(L_r) \cap S_0$ is $\mathcal{T}_r$-stable. We can compute $\mathcal{A}(L_l) \cap S_0$ and $\mathcal{A}(L_r) \cap S_0$ in $O(9^k n^2 + b_0)$ time, the values $M(\mathcal{T}_l(C_l))$ for all $C_l \in \mathcal{A}(L_l) \cap S_0$ in $T(|\mathcal{T}_l|, |\mathcal{S}_l|)$ time, and the values $M(\mathcal{T}_r(C_r))$ for all $C_r \in \mathcal{A}(L_r) \cap S_0$ in $T(|\mathcal{T}_r|, |\mathcal{S}_r|)$ time. Then, because any cell $C_0^j \in \mathcal{A}(L_0) \cap S_0$ is of the form $C_l \cap C_r$, with $C_l \in \mathcal{A}(L_l) \cap S_0$ and $C_r \in \mathcal{A}(L_r) \cap S_0$, we have $M(\mathcal{T}(C_0^j)) = M(\mathcal{T}_l(C_l)) + M(\mathcal{T}_r(C_r))$, and we can compute $M(\mathcal{T}(C_0^j))$ in constant time per cell. We conclude that it takes $O(9^k n^2)$ time for all cells in $\mathcal{A}(L_0) \cap S_0$.

The two cases give the recurrence

$$T(n, d) = O(9^k n^2 + b_0) + T(|\mathcal{T}_l|, |\mathcal{S}_l|) + T(|\mathcal{T}_r|, |\mathcal{S}_r|), \qquad T(n, 1) = O(9^k n^2),$$

where we still have freedom to choose the sets $\mathcal{S}_l$ and $\mathcal{S}_r$. The choice is made as follows. Assume without loss of generality that the subtree $\mathcal{T}_1$ is the biggest among the subtrees $\mathcal{T}_1, \dots, \mathcal{T}_d$, that is, $|\mathcal{T}_1| \geq |\mathcal{T}_i|$ for any $2 \leq i \leq d$. We distinguish two cases depending on the size of $\mathcal{T}_1$:

- If $|\mathcal{T}_1| \geq \frac{3n}{4}$, then $\mathcal{S}_l := \{S_1\}$ and $\mathcal{S}_r := \{S_2, \dots, S_d\}$.
- If $|\mathcal{T}_1| < \frac{3n}{4}$, we take $\mathcal{S}_l$ and $\mathcal{S}_r$ such that $\frac{n}{4} \leq |\mathcal{T}_l|, |\mathcal{T}_r| \leq \frac{3n}{4}$.

Taking $m = |\mathcal{T}_l|$ and $d' = |\mathcal{S}_l|$, we can rewrite the recurrence as

$$T(n, d) \leq \begin{cases} C9^k n^2 & \text{if } d = 1 \\ C(9^k n^2 + b_0) + T(m, 1) + T(n - m, d - 1) & \text{if } m \geq \frac{3n}{4} \text{ and } d > 1 \\ C(9^k n^2 + b_0) + T(m, d') + T(n - m, d - d') & \text{if } \frac{n}{4} < m, n - m < \frac{3n}{4} \text{ and } d > 1 \end{cases}$$

where $C > 0$ is some fixed constant. We will show by substitution that it solves to $T(n, d) \leq 3C(9^k n^2 + (d - 1)b_0) = O(9^k n^2 + db_0)$. Indeed, for the first case of the recurrence it is evident. For the second case we use $T(m, 1) \leq C9^k m^2$ to get

$$\begin{gathered} T(n, d) \leq C(9^k n^2 + b_0) + T(m, 1) + T(n - m, d - 1) \leq \\ \leq C(9^k n^2 + b_0) + C9^k m^2 + 3C(9^k (n - m)^2 + (d - 2)b_0) = \\ = C9^k (n^2 + m^2 + 3(n - m)^2) + Cb_0(1 + 3(d - 2)). \end{gathered}$$

Because $m \leq n$ and $n - m \leq n/4$ we have

$$T(n,d) \leq C9^k\big(n^2 + n^2 + 3(n/4)^2\big) + 3C(d-1)b_0 \leq C9^k(3n^2) + 3C(d-1)b_0.$$

For the third case we have

$$\begin{gathered} T(n,d) \leq C(9^k n^2 + b_0) + T(m,d') + T(n-m, d-d') \leq \\ \leq C\big(9^k n^2 + b_0 + 3(9^k m^2 + (d'-1)b_0) + 3(9^k(n-m)^2 + (d-d'-1)b_0)\big) \\ \leq C9^k\big(n^2 + 3m^2 + 3(n-m)^2\big) + Cb_0\big(1 + 3(d'-1) + 3(d-d'-1)\big). \end{gathered}$$

Because $m^2 + (n-m)^2$ is concave, and $n/4 < m, n-m < 3n/4$, we have

$$T(n,d) \leq C9^k\big(n^2 + 3(n/4)^2 + 3(3n/4)^2\big) + 3C(d-1)b_0 \leq C9^k(3n^2) + 3C(d-1)b_0$$

. □

Putting together Lemma 3 and Lemma 4 we can show how to compute $M(\mathcal{T})$ for a tree $\mathcal{T}$ of bounded height.

Lemma 5 *If each region S_i has complexity $O(n)$, and $\mathcal{T} = (\mathcal{S}, E)$ has height k, we can compute in $O(9^k n^2)$ time a placement $p_0, \ldots, p_{n-1}$ with $p_i \in S_i$ that achieves $M(\mathcal{T})$ alignments.*

Proof: Starting from the region S_0, we recursively apply the subdivision done in Lemma 2, and for the leaves, we take the whole region as a stable cell. For the leaves S_i we take $M(\mathcal{T}_i(S_i)) := 0$. Traversing the tree $\mathcal{T}$ in a bottom-to-top fashion, for each region S_i that has been subdivided into $\mathcal{T}_i$-stable cells $C_i^1, \ldots, C_i^{t_i}$ we compute all the values $M(\mathcal{T}_i(C_i^1)), \ldots, M(\mathcal{T}_i(C_i^{t_i}))$ using Lemma 4. Finally, we choose a cell $C_0^{j_0}$ such that $M(\mathcal{T}(C_0^{j_0})) = \max\{M(\mathcal{T}(C_0^1)), \ldots, M(\mathcal{T}(C_0^{t_0}))\}$, and a point $p_0 \in C_0^{j_0}$. If we have kept information on how we computed $M(\mathcal{T}(C_0^{j_0}))$ in Lemma 4, then it is easy to find points $p_1, \ldots, p_d$ such that

$$M(\mathcal{T}(p_0)) = \sum_{i=1}^{d} (\chi_O(p_0, p_i)) + M(\mathcal{T}_i(p_i))$$

and recursing on $M(\mathcal{T}_i(p_i))$ we get the placement for all points top-to-bottom.

Let b_i be the complexity of region S_i and let d_i be the degree of node S_i in $\mathcal{T}_i$. To bound the time needed, observe that for a node S_i that is at depth k_i, we have spent $O(9^{k-k_i}|\mathcal{T}_i|^2 + b_i)$ time to compute its subdivision into $\mathcal{T}_i$-stable cells $C_i^1, \ldots, C_i^{t_i}$ (Lemma 3), and $O(9^{k-k_i}|\mathcal{T}_i|^2 + d_i b_i)$ time to compute $M(\mathcal{T}_i(C_i^1)), \ldots, M(\mathcal{T}_i(C_i^{t_i}))$ (Lemma 4). To bound the time of the whole process, we sum over all nodes

$$\begin{gathered} \textstyle\sum_{S_i \in \mathcal{S}} O(9^{k-k_i}|\mathcal{T}_i|^2 + d_i b_i) = \sum_{k'=0}^{k} \Big(\sum_{S_i \text{ at depth } k'} O(9^{k-k'}|\mathcal{T}_i|^2)\Big) + \sum_{i=0}^{n-1} d_i b_i \leq \\ \leq \textstyle\sum_{k'=0}^{k} O\Big(9^{k-k'} \sum_{S_i \text{ at depth } k'} |\mathcal{T}_i|^2\Big) + O(n) \sum_{i=0}^{n-1} d_i \leq \\ \leq \textstyle\sum_{k'=0}^{k} O(9^{k-k'} n^2) + O(n^2) \leq O(9^k n^2). \end{gathered}$$

□

We can combine this last result with the shifting technique of Hochbaum and Maass [13] to get a polynomial time approximation scheme (PTAS) to approximate $M(\mathcal{T})$ for any tree $\mathcal{T}$, which is the main result of this section.

Theorem 3 *Let O be a set of two orientations, let $\mathcal{S} = \{S_0, \ldots, S_{n-1}\}$ be a set of n convex regions, each of complexity $O(n)$, and let $\mathcal{T} = (\mathcal{S}, E)$ be a tree. For any given integer $k > 0$, we can place points $p_0, \ldots, p_{n-1}$ with $p_i \in S_i$ that yield a $\frac{k}{k+1}$-approximation of $M_O(\mathcal{T})$ in $O(k9^k n^2)$ time.*

Proof: Choose any node S_0 of $\mathcal{T}$ to be the root. We apply the shifting technique of Hochbaum and Maass [13] in order to decompose the problem into trees of height k while controlling the loss in optimality. For $u = 0, \ldots, k$, consider the forest $\mathcal{F}_u$ that is obtained by removing from $\mathcal{T}$ the parent edge from any node that has distance $u + i \cdot (k+1)$ to the root node, where i is any integer. If we root each tree in $\mathcal{F}_u$ at the node that was closest to S_0 in $\mathcal{T}$, then it has height at most k, and we can use Lemma 5 to determine the optimum value $M_O(\mathcal{F}_u)$ in $O(9^k n^2)$ time. It is then clear that it takes $O(k9^k n^2)$ time to compute $M := \max\{M_O(\mathcal{F}_0), \ldots, M_O(\mathcal{F}_k)\}$.

We claim that M is a $\frac{k}{k+1}$-approximation of $M_O(\mathcal{T})$. To this end, consider the placement $p_0, \ldots, p_{n-1}$ with $p_i \in S_i$ that achieves $M_O(\mathcal{T})$ alignments. Because for each forest $\mathcal{F}_u$ we have $M_O(\mathcal{F}_u) \geq \sum_{\{S_i,S_j\}\in\mathcal{F}_u} \chi_O(p_i, p_j)$, then

$$(k+1)M \geq \sum_{u=0}^{k} M_O(\mathcal{F}_u) \geq \sum_{u=0}^{k} \sum_{\{S_i,S_j\}\in\mathcal{F}_u} \chi_O(p_i, p_j).$$

But if an edge is not in $\mathcal{F}_t$, then it is present in all $\mathcal{F}_u$ with $u \neq t$, and so each edge of $\mathcal{T}$ appears exactly k times in the sum. This means that

$$\sum_{u=0}^{k} \sum_{\{S_i,S_j\}\in\mathcal{F}_u} \chi_O(p_i, p_j) \geq k \sum_{\{S_i,S_j\}\in\mathcal{T}} \chi_O(p_i, p_j) = kM_O(\mathcal{T}),$$

and we conclude that M is a $\frac{k}{k+1}$-approximation of $M_O(\mathcal{T})$. □

Corollary 2 *Under the assumptions of the previous theorem, if $G = (\mathcal{S}, E)$ is a planar graph and $k > 0$ a given integer, we can compute a $\frac{k}{3(k+1)}$-approximation of $M_O(G)$ in $O(9^k n^2)$ time.*

5 Specific results for planar graphs

For different settings (regions and orientations) we can apply the same idea of dividing each region S_i into cells that are stable. The same recursive idea as explained before works out, but the analysis gives different results. Reconsidering Lemmas 3, 4, and 5 for each setting separately will give us the new bounds. We distinguish the following cases.

5.1 Any region, one orientation

Theorem 4 *Let O consist of one orientation, let $\mathcal{S} = \{S_0, \ldots, S_{n-1}\}$ be a set of n regions, and let $\mathcal{T} = (\mathcal{S}, E)$ be a tree. We can place points $p_0, \ldots, p_{n-1}$ with $p_i \in S_i$ that yield $M_O(\mathcal{T})$ in $O(n^2)$ time.*

Proof: We assume without loss of generality that the orientation for alignment to be considered is vertical. Also, as has been noted in the proof of Theorem 2, we can assume that the regions are horizontal segments, otherwise, we project each region onto a horizontal line and we get an equivalent problem.

In Lemma 3 we can get a more tight bound for $|L_0|$: in this setting each region produces two tangents (the vertical lines through its endpoints), and those are all the tangents that are created through the process, which means $|L_0| \leq 2n$. The lines L_0 induce a partition of the interval S_0 into $O(n)$ intervals, regardless of the height of $\mathcal{T}$. For Lemma 4, we can compute in a straightforward way the values $M(\mathcal{T}(C))$ in $O(d)$ time per cell C, where d is, as before, the degree of S_0. This means that we can accomplish Lemma 4 in $O(nd)$ time. For the time bound in Lemma 5, we have to sum over all nodes S_i the time spent at each node. If we denote by d_i the degree of S_i at $\mathcal{T}_i$ then we use

$$\sum_{i=0}^{n-1} O(|\mathcal{T}_i| d_i) \leq O\big(\sum_{i=0}^{n-1} n d_i\big) = O\big(n \sum_{i=0}^{n-1} d_i\big) = O(n^2)$$

time to accomplish Lemma 5. As this is independent of the height of $\mathcal{T}$, we directly get the statement. □

This, together with Lemma 1, leads to the following result.

Corollary 3 *Under the assumptions of the previous theorem, if $G = (\mathcal{S}, E)$ is a planar graph, we can get a $\frac{1}{3}$-approximation of $M_O(G)$ in $O(n^2)$ time.*

5.2 Axis-parallel rectangles, axis orientations

If the regions are disjoint rectangles, the placement of a point inside the region can be done independently for each axis orientation, and we can use the results from the previous subsection.

Corollary 4 *Let O be the orientations of the coordinate axes, let $\mathcal{S} = \{S_0, \ldots, S_{n-1}\}$ be a set of n axis-parallel rectangles, and let $G = (\mathcal{S}, E)$ be a planar graph. If the regions $\mathcal{S}$ are disjoint, we can get a $\frac{1}{3}$-approximation of $M_O(G)$ in $O(n^2)$ time.*

Proof: Consider the vertical orientation $o_v \in O$, and use Corollary 3 to compute a placement yielding a $\frac{1}{3}$-approximation of $M_{\{o_v\}}(G)$. This placement only fixes the x-coordinates of the points, and we can independently decide the y-coordinate of each point because the regions are rectangles. The y-coordinate is computed using Corollary 3 to get a $\frac{1}{3}$-approximation of $M_{\{o_h\}}(G)$, where $o_h \in O$ is the horizontal orientation. Because the regions are disjoint, no two points coincide and we have constructed a placement achieving at least $\frac{1}{3}(M_{\{o_v\}}(G) + M_{\{o_h\}}(G)) \geq \frac{1}{3}M_O(G)$ alignments. □

If the regions overlap, then this procedure only gives us a $\frac{1}{6}$-approximation because if points placed for different regions coincide, then we are counting them as two alignments. Without considering each orientation independently, but both as a whole, we can approximate this problem at the cost of another linear factor. Again, we have to consider first the case of a tree, and then combine it with Lemma 1 to approximate the planar graph case.

Theorem 5 *Let O be the orientations of the coordinate axes, let $\mathcal{S} = \{S_0, \ldots, S_{n-1}\}$ be a set of n axis-parallel rectangles, and let $\mathcal{T} = (\mathcal{S}, E)$ be a tree. We can place points $p_0, \ldots, p_{n-1}$ with $p_i \in S_i$ that yield $M_O(\mathcal{T})$ in $O(n^3)$ time.*

Proof: We reconsider Lemmas 3, 4, and 5 for this particular setting. In Lemma 3 we can get a more tight bound for $|L_0|$: in this setting each region produces four tangents (the axis-aligned lines containing the boundary of the region), and those are all the tangents that are created in the process, which means $|L_0| \leq 4n$. The lines L_0 induce a partition of the rectangle S_0 into $O(n^2)$ rectangles, regardless of the height of $\mathcal{T}$. For Lemma 4, we can compute in a straightforward way the values $M(\mathcal{T}(C))$ in $O(d)$ time per cell C, where d is, as before, the degree of S_0. This means that we can accomplish Lemma 4 in $O(n^2 d)$ time. For the time bound in Lemma 5, we have to sum over all nodes S_i the time spent at each node. If we denote by d_i the degree of S_i at $\mathcal{T}_i$ then we use

$$\sum_{i=0}^{n-1} O(|\mathcal{T}_i|^2 d_i) \leq O\big(\sum_{i=0}^{n-1} n^2 d_i\big) = O\big(n^2 \sum_{i=0}^{n-1} d_i\big) = O(n^3)$$

time to accomplish Lemma 5. As this is independent of the height of $\mathcal{T}$, we directly get the statement. □

Corollary 5 *Under the assumptions of the previous theorem, if $G = (\mathcal{S}, E)$ is a planar graph, we can get a $\frac{1}{3}$-approximation of $M_O(G)$ in $O(n^3)$ time.*

5.3 Convex regions, more than two orientations

We next assume that we are interested in alignment in $|O| > 2$ orientations ($|O|$ is a constant). For example, for schematic maps, alignment in the horizontal, vertical, and two diagonal orientations is important (thus $|O| = 4$). There are different approaches and corresponding results.

Corollary 6 *Let $G = (\mathcal{S}, E)$ be a planar graph with n nodes. We can find a $\frac{1}{3|O|}$-approximation of $M_O(G)$ in $O(n^2)$ time.*

Proof: We consider $|O|$ different problems, each one with a different orientation but with the same graph G. For each orientation $o \in O$, we use Corollary 3 to compute a $\frac{1}{3}$-approximation of $M_{\{o\}}(G)$ in $O(n^2)$ time, and we take the placement achieving the maximum A over all of them. Because in the placement achieving $M_O(G)$ alignments there is one orientation $\tilde{o} \in O$ with at least $\frac{1}{|O|}M_O(G)$ alignments, then for the chosen placement we get $A \geq \frac{1}{3}M_{\{\tilde{o}\}}(G) \geq \frac{1}{3|O|}M_O(G)$ alignments. □

Corollary 7 *For any integer $k > 0$, we can find a $\frac{2k}{3(k+1)|O|}$-approximation of $M_O(G)$ in $O(k9^k n^2)$ time.*

Proof: We consider $\binom{|O|}{2}$ different problems, each one with a different pair of orientations but with the same graph G. For each pair of orientations $\{o_i, o_j\} \subset O$, we use Corollary 2 to compute a $\frac{k}{3(k+1)}$-approximation of $M_{\{o_i,o_j\}}(G)$ in $O(k9^k n^2)$ time, and we take the placement achieving the maximum A over all of them. Because in the placement achieving $M_O(G)$ alignments there is a pair of orientations $\{\tilde{o}_i, \tilde{o}_j\} \subset O$ with at least $\frac{2}{|O|}M_O(G)$ alignments, then for the chosen placement we get $A \geq \frac{k}{3(k+1)}M_{\{\tilde{o}_i,\tilde{o}_j\}}(G) \geq \frac{2k}{3(k+1)|O|}M_O(G)$ alignments. □

Theorem 6 *Let O be a set of more than two orientations, let $\mathcal{S} = \{S_0, \ldots, S_{n-1}\}$ be a set of n convex regions, each of complexity $O(n)$, and let $G = (\mathcal{S}, E)$ be a planar graph. For any given integer $k > 0$, we can place points $p_0, \ldots, p_{n-1}$ with $p_i \in S_i$ that yield a $\frac{k}{3(k+1)}$-approximation of $M_O(G)$ in $n^{O(2^k)}$ time.*

Proof: We will show that for a tree $\mathcal{T} = (\mathcal{S}, E)$ of height k we can compute a $\frac{k}{k+1}$-approximation of $M_O(\mathcal{T})$ in $n^{O(2^k)}$ time. Then the proof of Theorem 3 implies that we can get a $\frac{k}{3(k+1)}$-approximation of $M_O(G)$ in $kn^{O(2^k)} \leq n^{O(2^k)+1} = n^{O(2^k)}$ time, as desired.

Let's assume that $\mathcal{T} = (\mathcal{S}, E)$ is a tree of height k, and reconsider Lemmas 3, 4, and 5 for this particular setting. The bound for $|L_0|$ in Lemma 3 is not longer true because each internal vertex produces $h - 2$ additional tangent lines (when we had only two orientations this did not happen). If for each child S_i of S_0, L_i is the set of lines that is used in the recursive process to subdivide the region S_i into $\mathcal{T}_i$-stable cells, then the lines in L_0 come from intersection points of two lines in L_i, from intersection points of a line in L_i with the boundary of the region S_i, and the $2|O|$ tangents to S_i itself. Taking $h = |O|$, the recursion that we get is

$$|L_0| \leq \sum_{i=1}^{d}\Big((h-2)\binom{|L_i|}{2} + (2h-1)|L_i| + 2h\Big) < 2h\sum_{i=1}^{d}(|L_i|+1)^2.$$

This solves to $|L_0| \leq (2h)^{(2^k-1)}n^{(2^k)} - 1$ by induction on the height k of $\mathcal{T}$: if $k = 0$, then $n = 1$ and it holds. For $k \geq 1$ we have

$$|L_0| \leq 2h\sum_{i=1}^{d}(|L_i|+1)^2 \leq 2h\sum_{i=1}^{d}\big((2h)^{(2^{k-1}-1)}|\mathcal{T}_i|^{(2^{k-1})}\big)^2,$$

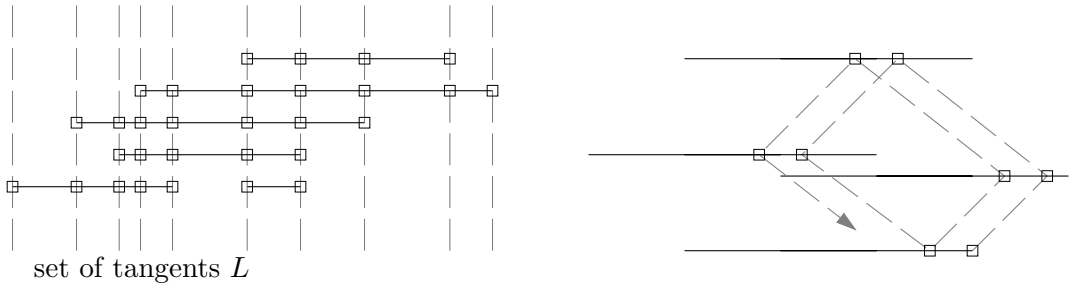

Figure 8: Left: Discretization used in the proof of Theorem 7. The boxes represent the only points to take into account. Right: If the regions are intervals and we consider two orientations to align points, we cannot discretize the problem.

and because $\sum_{i=1}^{d} |\mathcal{T}_i| = n - 1$ we get

$$|L_0| \leq (2h)^{(2^k-1)} \sum_{i=1}^{d} |\mathcal{T}_i|^{(2^k)} \leq (2h)^{(2^k-1)} (n-1)^{(2^k)}.$$

Therefore, $|L_0| = O((2h)^{(2^k-1)} n^{(2^k)}) = n^{O(2^k)}$ because we assume that $h = |O|$ is constant, and we can subdivide S_0 into $n^{O(2^k)}$ $\mathcal{T}$-stable cells in $O\big(\big(n^{O(2^k)}\big) + b_0\big) = n^{O(2^k)} + O(b_0)$ time.

For Lemma 4, we can compute in a straightforward way the values $M(\mathcal{T}(C))$ in $O(d)$ per cell C, where d is, as before, the degree of S_0. This means that we can accomplish Lemma 4 in $O(d)n^{O(2^k)} = n^{O(2^k)}$ time because $d \leq n$. For the time bound in Lemma 5, we have to sum over all nodes S_i the time spent in each node. If we denote by d_i the degree of S_i in $\mathcal{T}_i$, and because $b_i = O(n)$, we use

$$\sum_{i=0}^{n-1} (n^{O(2^k)} + b_i) \leq n^{O(2^k)+1} = n^{O(2^k)}$$

time to accomplish Lemma 5. □

6 Axis-aligned regions and orientations

We can use Baker's approach [2] for developing a polynomial time approximation schemes for planar graphs when we care about only one orientation.

Theorem 7 *Let O consist of one orientation, let $\mathcal{S} = \{S_0, \ldots, S_{n-1}\}$ be a set of n regions, and let $G = (\mathcal{S}, E)$ be a planar graph. For any given integer $k > 1$, we can place a points $p_0, \ldots, p_{n-1}$ with $p_i \in S_i$ that yield a $\frac{k-1}{k}$-approximation of $M_O(G)$ in $O(k(2n)^{3k+1})$ time.*

Proof: As in Theorem 4, it is enough to consider vertical alignments and regions that are horizontal segments. The proof goes in two steps. First, we show that for any k-outerplanar graph G, we can find a placement of points that attains the optimal solution $M_O(G)$ in $O((2n)^{3k+1})$ time. Second, we will show how this leads to the theorem.

Let L be the set of vertical lines going through the endpoints of the segments. Consider for each segment S_i the set of points $\tilde{S}_i := S_i \cap L$. Because L contains at most $2n$ vertical lines, $\tilde{S}_i$ consists of at most $2n$ points (see Figure 8, left). Now, instead of considering to place the point p_i anywhere in S_i, we want to place it at some point of $\tilde{S}_i$. In other words, if $\tilde{G}$ is the graph G, where each node S_i is replaced by $\tilde{S}_i$ (the graphs are isomorphic, but the nodes represent different sets), we have $M(G) = M(\tilde{G})$. Now that we have discretized the problem we can use Baker's approach.

Consider the slice boundaries and the slices as defined in [2] (we will follow its notation). In a level i slice boundary, we have at most $(2n)^i$ different ways of placing the points in the corresponding segments. Thus, for each level i slice, we can encode the maximum over all possible placements in its boundary using a table with at most $(2n)^{2i}$ entries. The operations between the tables are straightforward, and the most expensive one is merging two level i slices that share some level i boundary: it takes $O((2n)^{3i})$ time. If the graph is k-outerplanar, we have $i \leq k$, and we have to perform $O(n)$ operations with the tables. This concludes the first part of the proof.

For the given planar graph G and the integer $k > 0$, consider the graph G_u that we get by removing the edges connecting any level $u+ki$ vertex with a level $u+ki+1$ vertex, for all integers i. This graph G_u is composed of k-outerplanar graphs, so we can find the best placement of points as shown before. By the pigeon hole principle, there is some $u \in \{0, \ldots, k-1\}$ such that $M_O(G_u)$ is at least $\frac{k-1}{k} M(G)$. Computing all $M_O(G_u)$, for $u = 0, \ldots, k-1$, and taking the maximum leads to the result. □

Corollary 8 *For any set of orientations O, and for general regions, we can get a $\frac{k-1}{|O|k}$-approximation of $M_O(G)$ in $O(k(2n)^{3k+1})$ time.*

Proof: Like in the proof of Corollary 6, we can solve each orientation independently using the previous theorem and take the maximum over all of them. The result follows. □

As noticed in the proof of Corollary 4, if the regions are disjoint rectangles the placement of a point inside the region can be done independently for each axis orientation and we get the following result.

Corollary 9 *Let O be the orientations of the coordinate axes, let $\mathcal{S} = \{S_0, \ldots, S_{n-1}\}$ be a set of n axis-parallel rectangles, and let $G = (\mathcal{S}, E)$ be a planar graph. If the regions $\mathcal{S}$ are disjoint, for any integer $k > 1$ we can get a $\frac{k-1}{k}$-approximation of $M_O(G)$ in $O(k(2n)^{3k+1})$ time.*

When the rectangles overlap, we may be counting two alignments if two points belonging to different regions are placed exactly at the same position. We can avoid this adapting the proof of Theorem 7 to rectangles.

Theorem 8 *Let O be the orientations of the coordinate axes, let $\mathcal{S} = \{S_0, \ldots, S_{n-1}\}$ be a set of n axis-parallel rectangles, and let $G = (\mathcal{S}, E)$ be a planar graph. For any integer $k > 1$ we can get a $\frac{k-1}{k}$-approximation of $M_O(G)$ in $O(k(2n)^{6k+1})$ time.*

Proof: The proof is identical to the one of Theorem 7, but now, for solving a k-outerplanar graph we discretize the whole rectangles combining the vertical and horizontal lines (or tangents). This discretization uses $(2n)^2$ points per rectangle, and thus we can do it, by the same arguments as in that proof, in $O(k(2n)^{6k+1})$ time. □

For general regions or orientations, it doesn't seem easy to extend this approach. The problem is that we cannot discretize the problem as we have done before: each tangent can produce more candidate points, from which we have to trace new tangents, and this process doesn't converge (see Figure 8, right).

7 Remarks and conclusions

This paper studied algorithms to align points, each of which can be placed freely in their own specified region. Our motivation came from the automated computation of schematic networks for public transportation maps. We showed that the problem is computationally hard, and gave several approximation algorithms and approximation schemes which apply to different variations of the problem. Variations included the alignment orientations of interest, the shape of the regions,

whether overlap is allowed for any two regions, and perhaps most important, a graph on the points that specifies which alignments count in the optimization. Our results apply to trees and planar graphs, and remain valid if the edges of the graph are weighted. The problems and solutions gave rise to an interesting combination of geometry and graphs.

There is room to improve the results that we have presented. In particular, more tight results for the case of planar graphs, general regions, and general orientations would be a nice improvement. When the underlying graph is a tree, we have given a PTAS, but we do not know whether the problem is NP-hard or not. The answer to this question would not substantially improve the approximation factors for the case of planar graphs, but it is interesting in its own right.

Acknowledgements

The authors are grateful to Bettina Speckmann and Pankaj Agarwal for spending their time to discuss parts of this research.

References

[1] S. Avelar and M. Müller. Generating topologically correct schematic maps. In *Proc. 9th Int. Symp. on Spatial Data Handling*, pages 4a.28–4a.35, 2000.

[2] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41:153–180, 1994.

[3] T. Barbowsky, L.J. Latecki, and K. Richter. Schematizing maps: Simplification of geographic shape by discrete curve evolution. In *Spatial Cognition II, LNAI 1849*, pages 41–48, 2000.

[4] S. Cabello, M. de Berg, S. van Dijk, M. van Kreveld, and T. Strijk. Schematization of road networks. In *Proc. 17th Annu. ACM Sympos. Comput. Geom.*, pages 33–39, 2001.

[5] K. Daniels and V. J. Milenkovic. Multiple translational containment, part i: An approximate algorithm. *Algorithmica*, 19(1–2):148–182, September 1997.

[6] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, Germany, 2nd edition, 2000.

[7] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing*. Prentice Hall, Upper Saddle River, NJ, 1999.

[8] D. Elroi. Schematic views of networks: Why not have it all. In *Proc. of the 1991 GIS for Transportation Symposium*, pages 59–76. http://www.elroi.com/fr2_publications.html, 1991.

[9] H.N. Gabow. A matroid apporach to finding edge connectivity and packing arborescences. *J. Comput. Systems Sci.*, 50:259–273, 1995.

[10] R. Grossi and E. Lodi. Simple planar graph partition into three forests. *Discrete Applied Mathematics*, 84:121–132, 1998.

[11] J. Håstad. Some optimal inapproximability results. *J. ACM*, 48:798–859, 2001.

[12] R. Heckmann and T. Lengauer. Computing upper and lower bounds on textile nesting problems. In *Proc. 4th Annu. European Sympos. Algorithms*, volume 1136 of *Lecture Notes Comput. Sci.*, pages 392–405. Springer-Verlag, 1996.

[13] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *J. ACM*, 32:130–136, 1985.

[14] D. Lichtenstein. Planar formulae and their uses. *SIAM J. Comput.*, 11(2):329–343, 1982.

[15] G. Neyer. Line simplication with restricted orientations. In *Algorithms and Data Structures, WADS'99*, volume 1663 of *LNCS*, pages 13–24, 1999.

[16] J. O'Rourke. Visibility. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 25, pages 467–480. CRC Press LLC, Boca Raton, FL, 1997.

[17] M. Sharir. Algorithmic motion planning. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 40, pages 733–754. CRC Press LLC, Boca Raton, FL, 1997.

[18] E. Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer, editor, *New Results and New Trends in Computer Science*, volume 555 of *Lecture Notes Comput. Sci.*, pages 359–370. Springer-Verlag, 1991.

[19] A. Wolff and T. Strijk. The Map-Labeling Bibliography. http://www.math-inf.uni-greifswald.de/map-labeling/bibliography/, 1996.