Finding Maximally Consistent Sets of Halfspaces

Alon Efrat [*] Micha Lindenbaum [†] Micha Sharir [‡]

Abstract

Given a collection $\mathcal{S} = \{h_1, \dots h_n\}$ of n closed halfplanes in $\mathbb{R}^2$, we wish to find a subset $\mathcal{S}' \subseteq \mathcal{S}$ of maximum size whose intersection is nonempty. We show that solving this problem is a rather easy application of known techniques for constructing levels in arrangements of lines, and derive an $O(nk \log k + n\sqrt{k} \log n)$-time algorithm for solving this problem, where $k = n - |\mathcal{S}'| + 1$. A recent result of Erickson and Seidel [10] implies that finding a maximal intersecting subcollection among a given collection of n halfplanes takes $\Omega(n^2)$ time, and hence our bound is tight in the worst case, up to a logarithmic factor. We also consider a certain approximation to this problem, in which faster algorithms can be developed. We also discuss possible extensions of our method to higher dimensions and to other types of halfspaces, and present some applications from computer vision and pattern matching.

1 Introduction

Given a collection $\mathcal{S} = \{h_1, \dots h_n\}$ of n closed halfplanes, one wishes to solve the *maximally consistent subset* (MCS) problem with respect to $\mathcal{S}$, that is, to compute a subset of maximal size, so that all halfplanes in the subset have a point in common. Erickson and Seidel [10] recently showed that determining whether there exist 3 concurrent lines in a given planar collection of n lines requires time $\Omega(n^2)$ (in a somewhat restricted model of computation). By representing every line as the intersection of two infinitely close and opposite halfplanes, we get an arrangement of $2n$ halfplanes, in which there is a point contained in more than $n+3$ halfplanes if and only if 3 lines in the original arrangement intersect. Therefore, the Erickson-Seidel result implies that finding a point contained in a maximal number of halfplanes, or even finding whether there is a point included in $n+3$ halfplanes out of $2n$ given ones, requires quadratic time in the worst case (where essentially the only operation allowed on the given halfplanes is testing whether the point of intersection of the lines bounding two halfplanes lies inside a third halfplane).

On the other hand, determining whether n halfplanes have a point in common takes $O(n)$ time (using Megiddo's linear programing technique [12]). This raises the natural question of whether one can obtain faster algorithms in cases where it is known (or assumed) that most of the halfplanes do have a point in common. We show here that this is indeed the case, and present a simple algorithm for finding the maximally consistent subset (MCS), which is based on constructing the "at most k"-level in the arrangement of the halfplane boundaries. The algorithm provides the answer in time $O(nk \log k + n\sqrt{k} \log n)$, where $n-k+1$ is the maximal number of halplanes that do have a common point. Thus the algorithm is much faster than quadratic when k is small. We also propose an even simpler and faster algorithm for computing an approximate solution to the problem, in a sense defined below. We also discuss possible extensions of our technique to higher dimensions and to other types of halfspaces. We conclude the paper with a few applications of the algorithms to computer vision and pattern recognition.

2 The Algorithm

Consider the arrangement $\mathcal{A}$ formed by the n lines bounding the halfplanes in $\mathcal{S}$. The standard *level* of a point x in $\mathcal{A}$ is the number of lines passing below or through x. The *k-level* in $\mathcal{A}$ is the closure of all points at level k that lie on the edges of $\mathcal{A}$, but are not vertices (assuming general position, the vertices of $\mathcal{A}$ that belong to the k-level lie either at level k or at level $k+1$). Let $\mathcal{S}'$ be a maximum-size subset of

[*] School of Mathematical Sciences, Tel Aviv University

[†] Department of Computer Science, The Technion

[‡] School of Mathematical Sciences, Tel Aviv University and Courant Institute of Mathematical Sciences, New York University

$\mathcal{S}$ of halfplanes with a common point, and let $k^* = |\mathcal{S}| - |\mathcal{S}'|$. Before we describe the algorithm, it is worth mentioning that one always has $k^* \le \frac{n}{2}$, because one of the points at $y = +\infty$ or at $y = -\infty$ must lie in at least half of the given halfspaces (assuming there are no vertical halfplanes).

Lemma 1 *Partition $\mathcal{S}$ into the set $\mathcal{U}$ consisting of all halfplanes that contain the point $(0,\infty)$, and the complementary set $\mathcal{D} = \mathcal{S} \setminus \mathcal{U}$. Let $n_{\mathcal{U}} = |\mathcal{U}|$ and $n_{\mathcal{D}} = |\mathcal{D}|$. Let $L_i^{\mathcal{U}}$ be the i-level of the arrangement $\mathcal{A}(\mathcal{U})$ formed by the lines bounding the halfplanes in $\mathcal{U}$, and $L_i^{\mathcal{D}}$ be the $(n_{\mathcal{D}} - i)$-level of the arrangement $\mathcal{A}(\mathcal{D})$ formed by the lines bounding the halfplanes in $\mathcal{D}$. Then $k^* \ge k$ if and only if there are k_1, k_2, $k_1 + k_2 = k$, such that $L_{k_1}^{\mathcal{U}}$ and $L_{k_2}^{\mathcal{D}}$ intersect.*

Proof: If two such levels intersect at some point x, then, by definition, x lies in k_1 halfplanes of $\mathcal{U}$ and in k_2 halfplanes of $\mathcal{D}$, for a total of k halfplanes of $\mathcal{S}$. The converse assertion is proven in a similar manner. □

Guided by this lemma, we present the following simple algorithm. It consists of three phases. In the first phase we find a value k_1 that satisfies $k_1 = 2^{i-1} \le k^* < 2^i$. In the second phase we construct the "at-most-k_1" level of both arrangements $\mathcal{A}(\mathcal{U})$ and $\mathcal{A}(\mathcal{D})$. Formally, this is the portion of the arrangement, say $\mathcal{A}(\mathcal{U})$, consisting of all points at level at most k_1; in the second arrangement $\mathcal{A}(\mathcal{D})$ we define this region symmetrically, to consist of all points whose level is at least $n_{\mathcal{D}} - k_1$. Finally, we find in the third phase the exact size k^* of a maximally consistent subset, by overlaying these two regions and by checking the interaction between them.

2.1 First Phase: Finding k_1

Finding a value for k_1 that satisfies the above inequalities is easily done by unbounded search on the values of $k = 2^i$, for $i = 0, 1, \ldots$. For each value of k we construct the k-level of $\mathcal{U}$ and the $(n_{\mathcal{D}} - k)$-level of $\mathcal{D}$. If these levels do not intersect, then Lemma 1 implies that $k_1 = k$. Otherwise we double k and repeat this procedure. The construction is done using the output-sensitive technique of Cole et al. [5] (see also [8]), which computes the k-level in time $O(n \log n + b \log^2 k)$, where b is the complexity of the level, which is bounded by $O(n\sqrt{k})$ [7] (see also [16] for a slight improvement). Since a level is an x-monotone polygonal path, checking for intersection between two levels can be trivially done in time proportional to the combined complexity of both levels. The total cost of this phase is thus easily seen to be

$$O(n \log n \log k^* + n\sqrt{k^*} \log^2 k^*) .$$

2.2 Second Phase: Constructing the "at-most-k_1" Levels

The procedure used above for computing, say the k_1-level in the arrangement $\mathcal{A}(\mathcal{U})$ yields the intersection points of all the lines bounding halfplanes in $\mathcal{U}$ with that level. Since the complexity of the k-level is at most $O(n\sqrt{k})$ [7], these intersection points partition the lines of $\mathcal{U}$ into $O(n\sqrt{k_1}) = O(n\sqrt{k^*})$ segments, rays and full lines, and each of these pieces either lies fully below the k_1-level or fully above it. We take all these segments, rays, and lines that lie fully below the k_1-level and compute all their t intersection points, using the optimal algorithm of Chazelle and Edelsbrunner [3], or the randomized algorithm of Mulmuley [13]. Both algorithms provide the intersection points in the order in which they appear along each of the segments, making it straightforward to compute the level of each edge in the resulting arrangement. The time required by either algorithm (expected time for Mulmuley's algorithm) is $O(n\sqrt{k^*} \log n + t)$. However, as is well known, $t = O(nk_1) = O(nk^*)$ [7], so the running time of this phase is $O(n\sqrt{k^*} \log n + nk^*)$. Applying a symmetric procedure (replacing "below" by "above"), we also compute the "at-most-k_1" level in the arrangement $\mathcal{A}(\mathcal{D})$. Let us denote the two resulting subarrangements by $\mathcal{A}_{\le k_1}(\mathcal{U})$ and $\mathcal{A}_{\le k_1}(\mathcal{D})$, respectively.

2.3 Third Phase: Finding k^*

To find k^*, we have to overlap the two subarrangements $\mathcal{A}_{\le k_1}(\mathcal{U})$ and $\mathcal{A}_{\le k_1}(\mathcal{D})$, computed in the previous phase, to find a point whose sum of levels is as small as possible. Lemma 1 and the choice of k_1 guarantee that such a point does indeed lie in both these subarrangements. Moreover, we claim that it suffices to search for such a point only among the vertices of the two subarrangements (including also virtual points at infinity on the rays and full lines participating in these subarrangements). Indeed, if x is such a point, then (assuming general position) there is an entire face f in the superposition of these subarrangements, whose closure contains x and all of whose points lie in the same maximum number of halfspaces. But, as is easily verified, one of the vertices of f must be a vertex of one of the two subarrangements, thus establishing the claim.

We now merge the vertices of $\mathcal{A}_{\leq k_1}(\mathcal{U})$ and $\mathcal{A}_{\leq k_1}(\mathcal{D})$ into a single list Q sorted by x-coordinate. Since the vertices of each of these subarrangements can be represented as the union of at most k^* sorted lists (along each level separately), we can perform the merge in time $O(nk^* \log k^*)$. We next overlay these subarrangements by performing a standard line sweeping procedure over each subarrangement separately, using Q as the events queue of the sweep. This allows us to locate each vertex of one subarrangement in the other subarrangement. Moreover, since the size of the y-structure of the sweeps is always $O(k^*)$, the sweeps take a total of $O(nk^* \log k^*)$ time.

Summing up the bounds on the running time of the three phases of the algorithm, we see that the dominant terms are $O(nk^* \log k^* + n\sqrt{k^*} \log n)$, thus implying the main result of the paper:

Theorem 2 *The maximally consistent subset of n given halfplanes can be computed in time $O(n(k+1)\log(k+1) + n\sqrt{k+1}\log n)$, where k is the minimum number of halfplanes whose removal makes the remaining set consistent.*

Remark: We can speed up the algorithm if we only want an approximate value of k^*. That is, we only run the first phase of the algorithm, and get a value k_1 which approximates k^* to within a factor of 2. The running time is now only $O(n \log n \log k^* + n\sqrt{k^*} \log^2 k^*)$. Moreover, we can replace the factor 2 by any factor of the form $1 + \epsilon$, for any $\epsilon > 0$, by performing the same unbounded search as in Phase 1, over values of k of the form $(1+\epsilon)^i$, for $i = 0, 1, \ldots$. The running time becomes

$$O(\frac{1}{\epsilon} n \log n \log k^* + \frac{1}{\sqrt{\epsilon}} n\sqrt{k^*} \log^2 k^*) ,$$

where we also use the result of Welzl [19] concerning the overall complexity of several levels in an arrangement of lines.

3 Extensions

3.1 Generalization to higher dimensions

The maximally consistent subset problem can be extended naturally to $d \geq 3$ dimensions, where we have a collection $\mathcal{S}$ of n halfspaces, each bounded by a hyperplane, and we wish to find the subset of maximum size whose intersection is nonempty. The general outline of our algorithm can be extended in a straightforward manner, including an appropriate extension of Lemma 1. Unfortunately, efficient implementation of some of the steps of the algorithm is not available in higher dimensions.

In the first phase, instead of computing only a single level at a time, we compute the entire subarrangements $\mathcal{A}_{\leq k}(\mathcal{U})$, $\mathcal{A}_{\leq k}(\mathcal{D})$, and then test whether their boundaries (which are the desired k-levels) intersect. A recent randomized algorithm by Mulmuley [14] computes these subarrangements, and its expected time is $O(n^{\lfloor d/2 \rfloor} k^{\lceil d/2 \rceil})$ for $d \geq 4$ (which is worst-case optimal), and $O(nk^2 \log^2 \frac{n}{k})$ for $d = 3$ (which is almost worst-case optimal, up to a polylogarithmic factor). The main technical obstacle is in computing efficiently the superposition of these two subarrangements, which is required in the third phase of the algorithm.

In $d = 3$ dimensions, this can be done efficiently as follows. Each of these subarrangements consists of a collection of convex polytopes with pairwise disjoint interiors, whose overall combinatorial complexity is $O(nk^2)$ [4]. We preprocess each polytope for efficient ray shooting, using the technique of [6]. Then we start tracing the edges of, say $\mathcal{A}_{\leq k}(\mathcal{U})$, through the cells of the other subarrangement $\mathcal{A}_{\leq k}(\mathcal{D})$, and vice versa. Once we know the cell of $\mathcal{A}_{\leq k}(\mathcal{D})$ containing one endpoint of an edge e of $\mathcal{A}_{\leq k}(\mathcal{U})$, we start performing ray shooting along e, thereby finding all intersections of e with the faces of $\mathcal{A}_{\leq k}(\mathcal{D})$, in logarithmic time per intersection. The total number of intersections between edges of one subarrangement and faces of the other is also $O(nk^2)$, as follows easily from the results of [4, 17], so the total cost of the third phase of the algorithm is $O(nk^2 \log n)$. Filling in the missing details of the other phases of the algorithm, which we leave to the reader, one obtains:

Theorem 1 *The maximally consistent subset of n given halfspaces in 3-space can be computed in time $O(n(k+1)^2(\log n + \log^2 \frac{n}{k}))$, where k is the minimum number of halfspaces whose removal makes the remaining set consistent.*

3.2 Other Types of Halfspaces

Going back to $d = 2$ dimensions, we can generalize the MCS problem to cases where $\mathcal{S}$ is a given set of n halfplanes that are bounded by other types of closed or unbounded Jordan curves. This setup has been studied by Sharir [17], following the technique of [4], who gave sharp combinatorial bounds on the complexity of the "at-most-k" level in the arrangement of $\mathcal{S}$, namely the region consisting of all points that lie in at most k of the given halfplanes, or, symmetrically, in at least

$n-k$ of these halfplanes. It follows from the results of [17] that

(i) If each pair of the curves bounding the given halfplanes intersect at most twice (the case of *pseudo-discs*) then the complexity of the "at-most-k" level is $O(kn)$.

(ii) If the curves bounding the given halfplanes are all x-monotone and each pair of them intersect in at most s points (each halfplane lies either below or above such a curve), then the complexity of the "at-most-k" level is $O(k^2\lambda_s(n/k))$.

(iii) Using a simple divide-and-conquer algorithm, whose merge step consists of a standard line-sweeping procedure, the "at-most-k" level of the given arrangement can be computed in time $O(nk\log^2 n)$ in the case of pseudo-discs, and in time $O(k^2\lambda_s(n/k)\log^2 n)$ in the case of x-monotone curves.

These results suggest an algorithm similar to the one presented above. Its first phase also performs an unbounded search on $k = 2^i$, for $i = 0, 1, \ldots$, and tests for each such k whether the "at-most-k" level of the arrangement is empty or not, and then the second and third phases can be combined to a single phase that examines each vertex in the resulting subarrangement to find the one that lies in the maximum number of halfplanes. The total running time of the algorithm is easily seen to be $O(nk\log^2 n)$ in the case of pseudo-discs, and $O(k^2\lambda_s(n/k)\log^2 n)$ in the case of x-monotone curves.

4 Applications

Our problem can be formulated in the context of computer vision as a generalization of the *digital line problem*. There, one gets a set of n points on the grid (pixels), some of them black and some white, and has to decide whether this assignment pattern could follow from digitizing some straight edge (halfplane), that is, whether the two subsets are linearly separable. If this is the case, then the location of the straight edge can usually be found within subpixel accuracy, which is useful [2]. One way to solve this problem is to look at the dual problem, where every colored grid point is transformed to a halfplane in the plane of straight line parameters, and to ask whether all the halfplanes intersect, a question that can be answered in $O(n)$ time (see, for example, three on-line algorithms [15, 11, 18], of which the last two use the special properties of the grid). It turns out, however, that this approach is not useful for real images, where errors in the pixel color assignment may occur, and that methods which tolerate some errors are needed. The question of whether a set of pixels is consistent with some digitization pattern of a straight edge, excluding a finite number k of exceptions, is equivalent to asking whether $n-k$ of the dual halfplanes, in the plane of line parameters, intersect.

Another application is to detect and locate 2-dimensional elliptic point clusters in pattern recognition tasks. In a typical clustering task, one is given a mixed set of points on the plane, known to come from two different classes: one class (white points) that is believed to be scattered inside an elliptic region, except for a relatively small number of outliers, and another class (black points) that is just uniformly scattered. Due to the presence of outliers and the absence of an exact probabilistic model, it is difficult to find the ellipse using the traditional least squares minimization, and one just tries to find an elliptic plane region that will include as many of the white points as possible. This task can be solved by observing that if the ellipse contains a certain point, the coefficients (A, B, C, D, E, F) of its associated 2nd degree polynomial $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$ obey a certain linear inequality and must therefore lie in a certain halfspace of the 5-dimensional (projective) parameter space. Formally, let $\{(x_i, y_i),\ i \in \mathcal{I} \cup \mathcal{J}\}$ be a set of points, where $\{(x_i, y_i),\ i \in \mathcal{I}\}$ is the *white set*, and $\{(x_i, y_i),\ i \in \mathcal{J}\}$ is the *black set*. Map every point (x_i, y_i) to the vector

$$v_i = (1,\ x_i,\ y_i,\ x_i^2,\ y_i^2,\ x_i y_i)\,.$$

We seek a six-dimensional parameter vector c, whose first component is 1, such that the inequalities

$$v_i \cdot c \le 0 \quad i \in \mathcal{I} \quad \text{and} \quad v_i \cdot c > 0 \quad i \in \mathcal{J}$$

are satisfied for the maximal number of points. Geometrically, the inclusion of a certain white point in the ellipse corresponds to the parameters being in a parameter halfspace, and the absence of a black point corresponds to the parameters being in the opposite halfspace. By looking for the maximal intersection of halfspaces in the parameter space, we can find the parameters of the ellipse that includes the maximal number of white points and the minimal number of black points. To ensure that the parametrized second degree region is indeed an ellipse, we also need to ensure that the discriminant B^2-4AC is negative. Thus the problem reduces to the maximally consistent subset problem in 5-space, with the additional twist that the intersection of the halfspaces in a consistent subset is also required to meet a certain region bounded by the above quadric. Thus a full efficient solution to this problem still appears to be an open problem.

Another application is concerned with the exact positioning of objects with the aid of special figures painted on them. Such figures, called *fiducials*, are designed so that digitization of them will constrain their position with high precision. Although such fiducials are usually analyzed assuming an ideal noise-free context, we may use our approach to locate them even when errors are present. It was found that circles (and circular rings) perform very well as fiducials [1, 9]. Finding the parameters of such circles that are most consistent with the digitization pattern available, is just a special case of the problem of fitting the best ellipse, as described above, where now we want to fit the best circle. Here there are only three parameters that we have to determine (the general equation of a circle is $x^2 + y^2 + Ax + By + C = 0$), and there is no extra quadratic constraint on the parameters. Hence we can apply the 3-dimensional extension of our algorithm, and thus find the circle fitting most of the constraints in time $O(n(k+1)^2(\log n + \log^2 \frac{n}{k}))$, where k is the minimum number of outliers.

References

[1] A. Bruckstein, L. O'Gorman, and A. Orlitsky. Design of shapes for precise image registration. In *Technical report, AT&T*, 1991.

[2] C.A. Berenstein, L.N. Kanal, D. Lavine and E. Olson. A geometric approach to subpixel registration accuracy. *Comput. Vision, Graphics, and Image Processing*, 40(3):334–360 1987.

[3] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39:1–54, 1992.

[4] K. Clarkson and P. Shor, Applications of random sampling in computational geometry II, *Discrete Comput. Geom.* 4:387–421, 1989.

[5] R. Cole, M. Sharir and C.K. Yap, On k-hulls and related problems, *SIAM J. Computing* 16:61–77, 1987.

[6] D. Dobkin and D. Kirkpatrick, Fast detection of polyhedral intersection, *Theoretical Computer Science* 27:241–253, 1983.

[7] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer Verlag, Heidelberg, 1987.

[8] H. Edelsbrunner and E. Welzl, Constructing belts in 2-dimensional arrangements with applications, *SIAM J. Computing* 15:271–284, 1986.

[9] A. Efrat and C. Gotsman. Subpixel image registration using circular fiducials. In *The second Israel Symposium on Theory of Computing and Systems (ISTCS93)*, pages 47–48, 1993.

[10] J. Erickson and R. Seidel, Better lower bounds on detecting affine and spherical degeneracies, manuscript, 1993.

[11] M. Lindenbaum and A. Bruckstein. On recursive $o(n)$ partition of a digital curve into digital straight segments. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, to appear.

[12] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31:114–127, 1984.

[13] K. Mulmuley, A fast planar partition algorithm I, *J. Symbolic Computation* 10:253–280, 1990.

[14] K. Mulmuley, On levels in arrangements and Voronoi diagrams, *Discrete Comput. Geom.* 6:307–338, 1991.

[15] J. O'Rourke. An on-line algorithm for fitting straight lines between data ranges. *Commun. ACM*, 24:574–578, 1981.

[16] J. Pach, W. Steiger and E. Szemerédi, An upper bound on the number of planar k-sets, *Discrete Comput. Geom.* 7:109–123, 1992.

[17] M. Sharir, On k-sets in arrangements of curves and surfaces, *Discrete Comput. Geom.* 6:593–613, 1991.

[18] A. Smeulders and L. Dorst. Decomposition of discrete curves into piece-wise straight segments in linear time. *Vision Geometry, Melter, Rosenfeld and Bhattacharya, Eds., Contemporary Mathematics, AMS,*, 119:169–195, 1991.

[19] E. Welzl, More on k-sets of finite sets in the plane, *Discrete Comput. Geom.* 1:95–100, 1986.