SIAM J. COMPUT.
Vol. 12, No. 3, August 1983

0097-5397/83/1203-0006 $01.25/0

MULTIDIMENSIONAL SORTING*

JACOB E. GOODMAN† AND RICHARD POLLACK‡

Abstract. We introduce a process called *geometric sorting*, which can be applied to an arbitrary configuration of points in d-dimensional space, and which encodes in compact form the order properties of the configuration, just as the arrangement of a set of numbers in size place encodes its order properties. We give an algorithm for carrying out this sorting procedure in time $O(n^d \log n)$, which generalizes the optimum sorting time of $O(n \log n)$ for the linear case: In addition, we give an efficient algorithm for determining whether two randomly numbered configurations in $\mathbb{R}^d$ have the same order type, using a distinguished family of orderings of each. Finally, we indicate how this new concept of sorting can be applied to problems in pattern recognition, stereochemistry, and cluster analysis.

Key words. geometric sorting, computational geometry, efficient algorithms, planar configuration of points, multidimensional configuration, orientation, convex hull

Introduction. Finding the vertices of the convex hull of a given set of points in the plane, a problem that has been written about extensively in the last ten years [14], [19], [20], generalizes in a natural way the problem of determining the maximum and minimum of a given set of numbers on the real line. But there is much more to the "order type" of an indexed set $\{x_1, \cdots, x_n\}$ of numbers than just their maximum and minimum; this is why finding their extremes takes just $O(n)$ comparisons, while sorting them—by an optimal worst-case method such as a heapsort [18]—takes $O(n \log n)$ comparisons. In the same way, there is much more to a set of points in the plane—or in d-space, for that matter—than their convex hull. It is this more general problem of determing the "order type" of a given configuration of points in d-dimensional space, and determining it by a fast, efficient algorithm, to which this paper addresses itself.

If n numbers, $x_1, \cdots, x_n$, are given, there is just one kind of order pattern that they can have: that of the integers $1, 2, \cdots, r$ for some $r \leqq n$. The problem of sorting them comes down to finding a mapping π of the set $\{1, \cdots, n\}$ onto a set $\{1, \cdots, r\}$ that is "order-preserving", in the sense that $x_i \leqq x_j$ if and only if $\pi(i) \leqq \pi(j)$. But in the plane, or in higher dimensions, as we shall see, a configuration of n points can have many "patterns" or "order types". (It is a difficult problem, in fact, to find just how many, but we shall give some bounds below.) To sort such a configuration, we must therefore do *two* things: 1) determine which of the standard order patterns our configuration $\mathscr{C}$ fits, and 2) find a mapping from $\mathscr{C}$ into that standard pattern which exhibits the order isomorphism. We shall describe a means of carrying out each of these steps below.

First, what do we mean by the "order type" of a configuration of points? If n distinct points, $P_1, \cdots, P_n$, are given on a line, sorting them can be thought of in two ways: finding

(a) the set $\Lambda(i)$ of points on the positive side (i.e., to the right) of each P_i, since then the fundamental atoms of the ordering of $P_1, \cdots, P_n$, i.e. the ordered pairs i, j for which $P_i < P_j$, are immediately accessible, or

* Received by the editors September 30, 1981, and in revised form June 30, 1982.

† Department of Mathematics, City College of City University of New York, New York 10031. The work of this author was supported in part by the National Science Foundation under grant MCS 82-01831.

‡ Courant Institute of Mathematical Sciences, New York University, New York 10012. The work of this author was supported in part by the National Science Foundation under grant MCS 82-01342.

(b) the *number* $\lambda(i)$ of points on the positive side (i.e., to the right) of each P_i, since then the position of each point P_i in the ranking $P_{i_1} < \cdots < P_{i_n}$ is immediately deducible.

Furthermore, it is clear that from (b) we can read off (a), and of course (a) immediately gives (b). In the plane, indeed in d-dimensional space, each of these has a counterpart: the fundamental atoms of the "order pattern" made up of a planar configuration of points, what Dreiding and Wirth [6] call *chirons*, are the ordered triples i, j, k for which P_i, P_j, P_k are positively (i.e. "counterclockwise") oriented; hence to know these it would, as above, suffice to find

(a) the set $\Lambda(i, j)$ of points P_k lying on the positive side (i.e., to the left) of each directed line P_iP_j.

Similarly, the counterpart of (b) above would be to find

(b) the *number* $\lambda(i, j)$ of points P_k lying on the positive side (i.e., to the left) of each directed line P_iP_j.

It is a remarkable fact that—just as on the line—(a) and (b) are equivalent, in the sense that knowledge of one implies knowledge of the other. This result, which is not immediately obvious even in the planar case, was first shown, in that case, in [12], by the use of the method of "allowable sequences of permutations". Here we show by a direct geometric argument that it holds in every dimension, and even in the case where we allow degeneracies in our configurations: repeated points, collinear triples, coplanar quadruples, etc.; this is the substance of Theorem 1.8 below, which we call the "basic theorem of geometric sorting".

In §§ 2 and 3 we present a fast algorithm for sorting, i.e. determining the λ-function of, a configuration $\mathscr{C}$ of n points in $\mathbb{R}^d$; the d-dimensional algorithm in § 3 is based on the 2-dimensional one in § 2.

In §§ 4 and 5 we address the comparison problem: How do we determine whether two configurations of n points can be renumbered so that they are "λ-equivalent", i.e. have the same order type? Surprisingly enough, it turns out that we do not need to check all $n!$ renumberings, but at most n, in dimension 2, and $6n$-12, in dimension 3. In higher dimensions this number grows, but it is always bounded by $cn^{\lfloor d/2 \rfloor}$. In § 4 we introduce a distinguished family of orderings of the points of a configuration $\mathscr{C}$ in $\mathbb{R}^d$, which we call the "canonical orderings of $\mathscr{C}$", and which we believe to be of interest in their own right, in order to generate all the necessary renumberings. We present algorithms to handle various cases which may arise; in § 5 we present an algorithm for carrying out the entire comparison process.

Finally, in § 6 we discuss the optimality of the sorting algorithm, and we touch on several potential applications of geometric sorting: to pattern recognition, to stereochemistry, and to cluster analysis. We end with some open problems.

It should perhaps be pointed out that the notion of what we call the "order type" of a configuration $\mathscr{C}$ of points can be seen to be equivalent to what is known as the oriented matroid structure determined by $\mathscr{C}$ (see [2], [4], [5], [8]). However, its dual expression, in terms of Λ and λ, which forms the basis of this paper, is new here, as are the applications that follow to efficient methods of computing and encoding the order type of a configuration, and to the comparison problem for configurations.

1. The basic theorem of geometric sorting.

DEFINITION 1.1. If $(P_0, \cdots, P_d)$ is a sequence of $d+1$ points in $\mathbb{R}^d$, with $P_i = (x_{i1}, \cdots, x_{id})$ for each i, we say they have *positive orientation*, written $P_0 \cdots P_d > 0$, if

$$\det(x_{ij}) > 0,$$

where $x_{i0} = 1$ for each i. $P_0 \cdots P_d < 0$ and $P_0 \cdots P_d = 0$ are similarly defined.

Remark 1.2. If $d=1$, this reduces to: $P_0P_1>0$ whenever P_1 is to the "right" of P_0. If $d=2$, it says that $P_0P_1P_2>0$ whenever $P_0P_1P_2$ is oriented "counterclockwise", i.e., when P_2 is to the left of the directed line P_0P_1. And if $d=3$, it says that $P_0P_1P_2P_3>0$ whenever a "right-handed screw" turning in the direction $P_0P_1P_2$ moves toward P_3 (assuming the x, y, z-axes form a right-handed system!).

DEFINITION 1.3. If $\mathscr{C}=\{P_1,\cdots,P_n\}$ is a numbered configuration in $\mathbb{R}^d$ (with possible repetitions among the points P_i), and if $P_{i_0},\cdots,P_{i_{d-1}}$ affinely span a hyperplane of $\mathbb{R}^d$, let

$$\Lambda(i_0,\cdots,i_{d-1})=\{j|P_{i_0}\cdots P_{i_{d-1}}P_j>0\},$$

and let

$$\lambda(i_0,\cdots,i_{d-1})=|\Lambda(i_0,\cdots,i_{d-1})|;$$

if $P_{i_0},\cdots,P_{i_{d-1}}$ are affinely dependent, we write

$$\Lambda(i_0,\cdots,i_{d-1})=\Omega$$

and

$$\lambda(i_0,\cdots,i_{d-1})=\omega.$$

We can now define the "order type" of a numbered configuration $\mathscr{C}$ in $\mathbb{R}^d$ in two different ways, according to either the *number* $\lambda(i_0,\cdots,i_{d-1})$ of points on the positive side of each hyperplane $\langle P_{i_0},\cdots,P_{i_{d-1}}\rangle$, or the *set* $\Lambda(i_0,\cdots,i_{d-1})$ of these points. It is clear that if two configurations, $\mathscr{C}$ and $\mathscr{C}'$, are Λ-equivalent, in this sense, then they are λ-equivalent; Theorem 1.8 below shows that—just as on the line—the converse is true, i.e., knowing how *many* points lie on the positive side of each hyperplane $\langle P_{i_0},\cdots,P_{i_{d-1}}\rangle$ determines *which* ones do. First we need a few lemmas.

LEMMA 1.4. *If $P_1,\cdots,P_n$ affinely span $\mathbb{R}^d$, then $P_{i_0}=P_{i_1}$ if and only if $\lambda(i_0,i_1,j_2,\cdots,j_{d-1})=\omega$ for every choice of $j_2,\cdots,j_{d-1}$.*

Proof. The necessity of the condition is clear, and the sufficiency follows from the fact that the set consisting of any two distinct points can be completed to an affine spanning set of $d+1$ points.

LEMMA 1.5. *If $P_0,\cdots,P_d$ affinely span $\mathbb{R}^d$ and $\lambda(0,\cdots,d-1)=0$, then $\lambda(0,\cdots,j-1,d,j+1,\cdots,d-1)>0$ for each j, $0\leqq j\leqq d-1$.*

Proof. The hypothesis implies that $P_0\cdots P_d<0$. Hence $P_0\cdots P_{j-1}P_dP_{j+1}\cdots P_{d-1}P_j>0$, so that

$$j\in\Lambda(0,\cdots,j-1,d,j+1,\cdots,d-1).$$

LEMMA 1.6. *If $P_0\cdots P_d<0$ and if P' is any point lying in the hyperplane $\langle P_0,\cdots,P_{d-1}\rangle$, then $P_0,\cdots P_{j-1}P'P_{j+1}\cdots P_d<0$ for some j, $0\leqq j\leqq d-1$.*

Proof. Let $P_k=(x_{k1},\cdots,x_{kd})$ for each k. When we write

$$\overrightarrow{OP'}=\sum_{0\leqq k\leqq d-1}c_k\overrightarrow{OP_k},\qquad\text{with}\sum_k c_k=1,$$

we must have $c_j>0$ for some j, $0\leq j\leq d-1$. The result then follows by splitting the

determinant

$$\begin{vmatrix} 1 & x_{01} & \cdots & x_{0d} \\ \vdots & \vdots & & \vdots \\ 1 & x_{j-1,1} & \cdots & x_{j-1,d} \\ \sum c_k & \sum c_k x_{k1} & \cdots & \sum c_k x_{kd} \\ 1 & x_{j+1,1} & \cdots & x_{j+1,d} \\ \vdots & \vdots & & \vdots \\ 1 & x_{d1} & \cdots & x_{dd} \end{vmatrix}$$

into a sum of determinants along its jth row by multilinearity, and noting that all the summands give 0 except for the jth, which gives

$$c_j \begin{vmatrix} 1 & x_{01} & \cdots & x_{0d} \\ & & \cdots & \\ 1 & x_{d1} & \cdots & x_{dd} \end{vmatrix},$$

and this is negative.

LEMMA 1.7. *Let H be a hyperplane in $\mathbb{R}^d$, let P_0 be any point not in H, and let H_0 be the hyperplane parallel to H and passing through P_0. Finally, let U be the halfspace of $\mathbb{R}^d$ that is on the same side of H_0 as H, i.e., the component of $\mathbb{R}^d \backslash H_0$ containing H, and let $\pi : U \to H$ denote projection from P_0. Then one of the two orientations of H as a $(d-1)$-dimensional affine space in its own right will agree with the standard orientation of $\mathbb{R}^d$ in the following sense: if $P_1, P_2, \cdots, P_d$ are points of U, then $P_0P_1 \cdots P_d$ will be positively oriented, negatively oriented, or affinely dependent in $\mathbb{R}^d$ if and only if $\pi(P_1)\pi(P_2) \cdots \pi(P_d)$ has the corresponding property in H.*

Proof. Suppose $P_0P_1 \cdots P_d > 0$ for some choice of $P_1, \cdots, P_d$ in U. H has only two orientations. Let us fix one in which $\pi(P_1) \cdots \pi(P_d) > 0$ also. Then if $Q_1, \cdots, Q_d$ are any other points in U for which $P_0Q_1 \cdots Q_d > 0$, we can move the simplex $P_0, P_1, \cdots, P_d$ continuously to $P_0, Q_1, \cdots, Q_d$ (with P_0 remaining fixed and the other vertices remaining in U) so that each intermediate configuration also has positive orientation. Since $\pi(R_1), \cdots, \pi(R_d)$ are affinely dependent in H if and only if $P_0, R_1, \cdots, R_d$ are affinely dependent in $\mathbb{R}^d$, it follows by the intermediate value theorem that we must have $\pi(Q_1) \cdots \pi(Q_d) > 0$ as well, from which the assertion follows immediately.

THEOREM 1.8. *If, for a configuration $\mathscr{C} = \{P_1, \cdots, P_n\}$ in $\mathbb{R}^d$, the function λ is given, then the function Λ is uniquely determined.*

Proof. The proof is by double induction: by induction on d and, for a given d, by induction on n. Let us first note that the statement is clear for $d = 1$, as is immediate from Remark 1.2. Suppose, then, that $d > 1$. We may assume $\lambda(i_0, \cdots, i_{d-1}) > 0$ for some $i_0, \cdots, i_{d-1}$; otherwise the statement is trivially true. Now let us fix $i_0, \cdots, i_{d-1}$ such that $\lambda(i_0, \cdots, i_{d-1}) = 0$; such a d-tuple must exist, since the convex hull Δ of $\{P_1, \cdots, P_n\}$ has faces that span hyperplanes, and—if necessary, by interchanging two of the indices—we may select a spanning set for such a hyperplane that has no points of $\mathscr{C}$ on its "positive" side. Let F be the face of Δ on which the points $P_{i_0}, \cdots, P_{i_{d-1}}$ lie. If k is any index such that $\lambda(i_0, \cdots, i_{j-1}, k, i_{j+1}, \cdots, i_{d-1}) = 0$ for some j, $0 \leqq j \leqq d-1$, then the points $P_{i_0}, \cdots, P_{i_{j-1}}, P_k, P_{i_{j+1}}, \cdots, P_{i_{d-1}}$ also lie on a face F' of Δ, and we must have $F' = F$: otherwise the result of Lemma 1.5 would be violated; hence $P_k \in F$. Conversely, if P_k is any point on F we must have $\lambda(i_0, \cdots, i_{j-1}, k, i_{j+1}, \cdots, i_{d-1}) = 0$ for some j; this follows from Lemma 1.6 since the points of $\mathscr{C}$ span $\mathbb{R}^d$. Thus λ determines all of the points lying on each face of Δ. Now

by Lemma 1.4, λ determines the equivalence classes of the relation

$$i \sim j \Leftrightarrow P_i = P_j;$$

we can therefore coalesce each "cluster" of repeated points into a single point, and get a configuration $\bar{\mathscr{C}}$ consisting of distinct points $\bar{P}_1, \cdots, \bar{P}_r$, with $r \leqq n$. In particular, we know the faces of the convex hull $\bar{\Delta}$ of $\bar{\mathscr{C}}$—they are just the images of the faces of Δ. Hence, since an extreme point of $\bar{\Delta}$ is an intersection of faces that has cardinality 1, we can identify the extreme points of $\bar{\Delta}$, and so also the extreme points of Δ. Say P_n is such an extreme point, and say $\{P_{m+1}, \cdots, P_n\}$ is the cluster of repeated points to which P_n belongs. Let H_0 be a supporting hyperplane of Δ at P_n, and let U be the open halfspace determined by H_0 which contains the points $P_1, \cdots, P_m$. Let H be a hyperplane parallel to H_0 and lying in U, with the orientation given by Lemma 1.7; as in that lemma, let $\pi : U \to H$ be projection from P_n. Then by that lemma, if we let Λn (resp. λ_n) be the Λ- (resp. λ-) function for the configuration $\mathscr{C}_n = \{\pi(P_1), \cdots, \pi(P_m)\}$, we have

$$\lambda_n(\pi(P_{i_1}), \cdots, \pi(P_{i_{d-1}})) = \lambda(P_n, P_{i_1}, \cdots, P_{i_{d-1}}),$$

for any choice of indices. This shows that the function λ_n is determined, so—by induction on the dimension—Λ_n is also. But this means, again by Lemma 1.7, that we can determine the orientation of every $d+1$-tuple of $\mathscr{C}$ involving P_n, i.e.,

(1.1) *we can identify those d-tuples* $P_{i_0}, \cdots, P_{i_{d-1}} \in \mathscr{C}$ *for which* $P_{i_0} \cdots P_{i_{d-1}} P_n > 0$.

Let $\tilde{\mathscr{C}}$ be the configuration $\{P_1, \cdots, P_m\}$ that remains when the entire cluster of repeated points to which P_n belongs has been deleted; then we have just shown that the λ-function $\tilde{\lambda}$ of $\tilde{\mathscr{C}}$ is determined: it is given by

$$\tilde{\lambda}(i_0, \cdots, i_{d-1}) = \lambda(i_0, \cdots, i_{d-1}) \qquad (\text{resp. } \lambda(i_0, \cdots, i_{d-1}) - (n-m))$$
$$\text{if } P_{i_0} \cdots P_{i_{d-1}} P_n \leqq 0 \qquad (\text{resp.} > 0).$$

Hence by induction on the number of points of the configuration, the corresponding function $\tilde{\Lambda}$ is also determined, and therefore—again with the help of (1.1)—so is Λ, which proves the theorem.

COROLLARY 1.9. *If two configurations have the same λ-function, they have the same order type, i.e., the same orientation of $(d+1)$-tuples $P_{i_0}, \cdots, P_{i_d}$ for every $i_0, \cdots, i_d$.*

We can therefore formalize the notion of order type by stating

DEFINITION 1.10. If $\mathscr{C} = \{P_1, \cdots, P_n\}$ and $\mathscr{C}' = \{P_1', \cdots, P_n'\}$ are numbered configurations in $\mathbb{R}^d$ for which $\lambda_{\mathscr{C}}(i_0, \cdots, i_{d-1}) = \lambda_{\mathscr{C}'}(i_0, \cdots, i_{d-1})$ for every $i_0, \cdots, i_{d-1}$, we say $\mathscr{C}$ and $\mathscr{C}'$ have the same order type, or $\mathscr{C}$ is equivalent to $\mathscr{C}'$ ($\mathscr{C} \sim \mathscr{C}'$).

By analogy with the interpretation [18, p. 4] of "sorting" a set of numbers as meaning determining the position of each when they are arranged in increasing order, we then have:

DEFINITION 1.11. If points $P_1, \cdots, P_n$ in $\mathbb{R}^d$ are given, to *sort* them *geometrically* will mean to determine the associated λ-function.

Remark 1.12. If $P_1, \cdots, P_n$ affinely span $\mathbb{R}^d$, the sorting function λ determines not only the "d-dimensional order type" of any subset, but also the linear order of a collinear subset, the "2-dimensional order type" of a coplanar subset, and so on—just fill out the remaining places with points chosen in general position, and use the result of Theorem 1.8. If $P_1, \cdots, P_n$ do *not* span $\mathbb{R}^d$, however, then sorting them in the sense above yields very little information on their actual (lower-dimensional) order

type, since the value of λ is either 0 or ω for each d-tuple. But in this case, by throwing in the points $(1, 0, \cdots, 0), (0, 1, 0, \cdots, 0), \cdots, (0, \cdots, 0, 1)$ and *then* sorting, we can get all the information about the order type of the original configuration. From now on, whenever we speak of sorting a configuration $\mathscr{C}$ in $\mathbb{R}^d$, we shall therefore assume that $\mathbb{R}^d$ is spanned by the points of $\mathscr{C}$.

2. An algorithm for sorting in the plane. Here is a fast algorithm for determining the λ-matrix of a planar configuration. The idea is quite simple: consider each point P_i, in turn, as an "origin", reflect each $P_j \neq P_i$ in P_i, obtaining new points "P_{n+j}", and sort the points and their reflections circularly about P_i, the order within each ray about P_i being immaterial. In order to obtain $\lambda(i, j)$, we then have merely to count the number of original points from the ray P_iP_j around to the ray P_iP_{n+j}.

ALGORITHM 2.1.

Input: $(P_i = (x_i, y_i))$, $1 \leqq i \leqq n$.

Output: $(\lambda(i, j))$, $1 \leqq i \leqq n$, $1 \leqq j \leqq n$.

1. (Starting with $i = 1$) pick the next i (ending with $i = n$).
2. For every $j = 1, \cdots, n$, let $u_i = x_j - x_i$, $v_j = y_j - y_i$.
3. For every $j = 1, \cdots, n$, if $(u_j, v_j) = (0, 0)$, let $\lambda(i, j) = \omega$; otherwise, call j "good".
4. For every good $j = 1, \cdots, n$, let $u_{n+j} = -u_j$, $v_{n+j} = -v_j$, and let $m_j = m_{n+j} = v_j/u_j$ if $u_j \neq 0$.
5. Sort the indices $\{j \mid j \text{ is good}\} \cup \{n + j \mid j \text{ is good}\}$ into subsets, as follows:
   (a) first those for which $u_j > 0$, using m_j as key;
   (b) next those for which $u_j = 0$ and $v_j > 0$;
   (c) next those for which $u_j < 0$, using m_j as key;
   (d) finally those for which $u_j = 0$ and $v_j < 0$.

(In (a) and (c) we get a list of subsets; the order within each subset is irrelevant.) Say the result is

$$J_{11}, \cdots, J_{1s_1}, \cdots, J_{r1}, \cdots, J_{rs_r},$$

where the points with indices $J_{k1}, \cdots, J_{ks_k}$ constitute an entire subset, and there are r subsets all together.

6. For each $k = 1, \cdots, r$, let

$$n_k = |\{m \mid 1 \leqq m \leqq s_k, J_{km} \leqq n\}|.$$

7. For each good $j = 1, \cdots, n$, let

$$\lambda(i, j) = \begin{cases} \displaystyle\sum_{k=k(j)+1}^{k(j+n)-1} n_k & \text{if } k(j+n) > k(j), \\ \displaystyle\sum_{k=k(j)+1}^{r} n_k + \sum_{k=1}^{k(j+n)-1} n_k & \text{if } k(j+n) < k(j). \end{cases}$$

(Note: $k(j)$ means, of course, the number of the subset within which j lies.)

8. Return to step 1.

Analysis. Step 2 translates each point P_i, in turn, to the origin O. Step 3 calls λ "undefined" if $P_j = P_i$, and designates P_j as "good" otherwise. Step 4 reflects each good P_j in P_i, and calculates the slope of P_iP_j. Step 5 sorts the good points and their reflections into rays starting at O, listed in counterclockwise order, the order along each ray being immaterial (see Fig. 1). Step 6 counts the *original* points in each ray. Finally, step 7 counts the number of original points going counterclockwise from the ray containing P_j to the ray containing its reflection: this gives $\lambda(i, j)$.

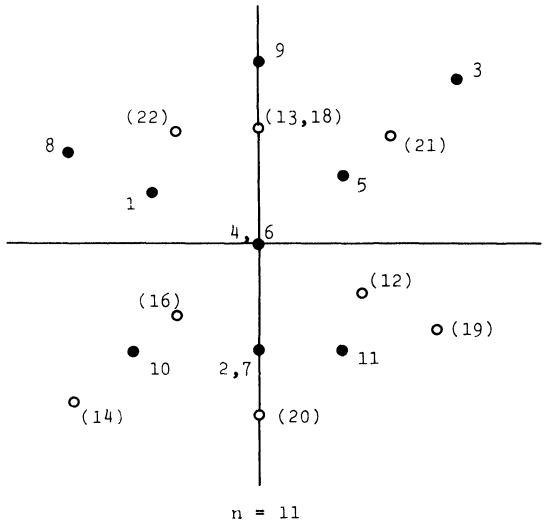


FIG. 1. $n = 11; J_{11}, \cdots, J_{83} = 11; 12, 19; 3, 5, 21; 9, 13, 18; 22; 1, 8; 10, 14, 16; 2, 7, 20.$

For each $i = 1, \cdots, n$, steps 2, 3, 4, and 6 take time $O(n)$, step 7 also takes time $O(n)$ if the sum for each j is computed by correcting the sum for the previous j, and the sorting in step 5 takes $O(n \log n)$ if done by an optimal worst-case method, such as heapsort [18]; hence for each i the time is $O(n \log n)$. It follows that the total sorting time is $O(n^2 \log n)$.

3. An algorithm for sorting in higher dimensions. The process of geometric sorting is not quite so simple in higher dimensions as in the plane, if one wants to obtain an efficient algorithm. For example, a straightforward approach would be to project the configuration $\mathscr{C}$ parallel to each hyperplane $\langle P_{1_i}, \cdots, P_{i_d}\rangle$ onto a coordinate axis transversal to that hyperplane, and simply count the number of points on each side of the projection of the hyperplane itself. This, however, would take time $O(n^{d+1})$ for fixed d, and this is precisely what we are trying to avoid, since the λ-function for a configuration in $\mathbb{R}^d$ occupies only $O(n^d)$ space.

It turns out that a combination of this projection method (but onto a *plane* instead of a line), together with the sorting-by-slope procedure of Algorithm 2.1, reduces the time to $O(n^d \log n)$. Here is the basic idea (the details, which are a bit complicated, are presented in the Analysis section immediately following the algorithm): For each choice of $d-1$ points in general position, we choose a coordinate plane Π transversal to their span Σ, and project the entire configuration parallel to Σ onto Π. We then sort the projected points and their reflections circularly around the projection of Σ, precisely as in Algorithm 2.1, and the same device used there, of counting around from a ray through a point to the ray through its reflection, then gives the value of λ. A few "tricks" are used in the algorithm to shorten the execution time. These include (a) a dilatation of the projected configuration around the "origin", which comes from suppressing the denominators in the application of Cramer's rule used to find the projected configuration, and which has no effect on the order type of the projection, and (b) finding $\lambda(i_1, \cdots, i_d)$ only for $i_1 < \cdots < i_{d-2} < \{i_{d-1}, i_d\}$ and then applying even permutations to the indices to find the remaining values of λ; these are explained in more detail in the Analysis section below.

ALGORITHM 3.1′.

Input: $(P_i = (x_{i1}, \cdots, x_{id}))$, $1 \leqq i \leqq n$.

Output: $(\lambda(i_1, \cdots, i_d))$, $1 \leqq i_1, \cdots, i_d \leqq n$.

1. Let $\lambda(i_1, \cdots, i_d) = \omega$ for every $i_1, \cdots, i_d$, $1 \leqq i_j \leqq n$.

2. (Starting with $i_1, \cdots, i_{d-1} = 1, \cdots, d-1$) pick the next $i_1, \cdots, i_{d-1}$ with $i_1 < \cdots < i_{d-1}$ (ending with $n-d+2, \cdots, n$).

3. If

$$\det \begin{pmatrix} 1 & x_{i_1 1} & \cdots & x_{i_1 j-1} & x_{i_1 j+1} & \cdots & x_{i_1 k-1} & x_{i_1 k+1} & \cdots & x_{i_1 d} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{i_{d-1} 1} & \cdots & x_{i_{d-1} j-1} & x_{i_{d-1} j+1} & \cdots & x_{i_{d-1} k-1} & x_{i_{d-1} k+1} & \cdots & x_{i_{d-1} d} \end{pmatrix}$$

vanishes for every j, k with $1 \leqq j < k \leqq d$, go to step 2.

Otherwise, let (α, β) be the first (j, k) for which this determinant $\neq 0$, and let $\sigma = \pm 1$ be the sign of this determinant.

4. For each $j = 1, \cdots, n$, let

$$y_\alpha(j) = \det \begin{pmatrix} 1 & x_{i_1 1} & \cdots & x_{i_1, \beta-1} & x_{i_1, \beta+1} & \cdots & x_{i_1 d} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{i_{d-1} 1} & \cdots & x_{i_{d-1}, \beta-1} & x_{i_{d-1}, \beta+1} & \cdots & x_{i_{d-1} d} \\ 0 & x_{j1} & \cdots & x_{j, \beta-1} & x_{j, \beta+1} & \cdots & x_{jd} \end{pmatrix},$$

$$y_\beta(j) = \det \begin{pmatrix} 1 & x_{i_1 1} & \cdots & x_{i_1, \alpha-1} & x_{i_1, \alpha+1} & \cdots & x_{i_1 d} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{i_{d-1} 1} & \cdots & x_{i_{d-1}, \alpha-1} & x_{i_{d-1}, \alpha+1} & \cdots & x_{i_{d-1} d} \\ 0 & x_{j1} & \cdots & x_{j, \alpha-1} & x_{j, \alpha+1} & \cdots & x_{jd} \end{pmatrix}.$$

5. If $\sigma = +1$, let $u'(j) = y_\alpha(j)$ and $v'(j) = y_\beta(j)$ for each $j = 1, \cdots, n$; if $\sigma = -1$, let $u'(j) = y_\beta(j)$ and $v'(j) = y_\alpha(j)$ for each $j = 1, \cdots, n$.

6. For each $j = 1, \cdots, n$, let $u_j = u'(j) - u'(i_1)$ and $v_j = v'(j) - v'(i_1)$.

Now perform steps 3 through 7 of Algorithm 2.1, modified as follows:

7. For each $j = 1, \cdots, n$, if $(u_j, v_j) \neq (0, 0)$, call j "good".

8. Same as step 4, but define u_{n+j}, v_{n+j}, and m_{n+j} only for good $j = i_{d-2}+1, \cdots, n$; define m_j for *all* good $j = 1, \cdots, n$.

9. Same as step 5, but only for the indices $\{j \mid j \text{ is good}\} \cup \{n+j \mid j \text{ is good and } j \geqq i_{d-2}+1\}$.

10. Same as step 6.

11. Same as step 7, but only for good $j = i_{d-2}+1, \cdots, n$; call the result $\lambda(i_1, \cdots, i_{d-1}, j)$.

12. For each good $j = i_{d-1}+1, \cdots, n$, apply every even permutation π, in turn, to $i_1, \cdots, i_{d-1}, j$; let

$$\lambda(\pi(i_1), \cdots, \pi(i_{d-1}), \pi(j)) = \lambda(i_1, \cdots, i_{d-1}, j).$$

13. Return to step 2.

Analysis. Algorithm 3.1 computes $\lambda(i_1, \cdots, i_d)$ whenever $i_1 < \cdots < i_{d-2} < \{i_{d-1}, i_d\}$, provided the points $P_{i_1}, \cdots, P_{i_{d-1}}$ are in general position, and then applies even permutations to these ordered d-tuples of indices and records the same values for λ. In this way *all* ordered d-tuples for which λ is defined have λ computed for them. In all other cases, where λ is undefined, we write $\lambda = \omega$ to represent this. It is therefore easier to write $\lambda = \omega$ everywhere to begin with, and write over this whenever the need arises; hence step 1. The determinant (call it Δ) computed in step 3 serves

three purposes: (a) it determines whether the points $P_{i_1}, \cdots, P_{i_{d-1}}$ span a $(d-2)$-flat, (b) it identifies a coordinate plane (the x_α, x_β-plane) transversal to that $(d-2)$-flat, and (c) its sign reflects the orientation of the ordered $(d-1)$-tuple $P_{i_1}, \cdots, P_{i_{d-1}}$. Step 4 comes from an application of Cramer's rule, used to find the projection $f(P_j)$ of each point P_j parallel to the $(d-2)$-flat $\langle P_{i_1}, \cdots, P_{i_{d-1}}\rangle$ into the x_α, x_β-plane. We have suppressed the denominators that Cramer's rule gives; doing so amounts to a dilatation around the origin, and possibly to a reflection through the origin as well, if the suppressed denominator is negative. This simplifies the calculation, and neither the dilatation nor the reflection affects the collinearity or the orientation of points in the x_α, x_β-plane. Step 5 compensates for the orientation of the points $P_{i_1}, \cdots, P_{i_{d-1}}$; the result of steps 4 and 5 is that the points $P_{i_1}, \cdots, P_{i_{d-1}}, P_j, P_k$ have positive orientation in $\mathbb{R}^d$ if and only if $f(P_{i_1}), f(P_j), f(P_k)$ have positive orientation in the u, v-plane (with the coordinates in their natural order in both cases). This is most easily seen as follows: Let us first check it in the special case of the points $P_0(0, \cdots, 0)$, $P_1(1, 0, \cdots, 0), \cdots, P_d(0, \cdots, 0, 1)$. Choose any α, β with $1 \leqq \alpha < \beta \leqq d$, and let π be any permutation of $\{0, \cdots, d\}$ such that $\pi(d-1) = \alpha$, $\pi(d) = \beta$. Let

$$\bar{\Delta} = \det \begin{pmatrix} 1 & x_{\pi(0),1} & \cdots & x_{\pi(0),d} \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_{\pi(d),1} & \cdots & x_{\pi(d),d} \end{pmatrix};$$

we have $\bar{\Delta} = \operatorname{sgn}(\pi)$, of course. On the other hand, the $(d-2)$-flat along which we are going to project to the x_α, x_β-plane is $\langle P_{\pi(0)}, \cdots, P_{\pi(d-2)}\rangle$, so it is precisely the αth and βth columns we must leave out to get a nonzero determinant; hence

$$\Delta = \det \begin{pmatrix} 1 & x_{\pi(0)1} & \cdots & x_{\pi(0),\alpha-1} & x_{\pi(0),\alpha+1} & \cdots & x_{\pi(0),\beta-1} & x_{\pi(0),\beta+1} & \cdots & x_{\pi(0)d} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{\pi(d-2)1} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & x_{\pi(d-2)d} \end{pmatrix}.$$

Expanding $\bar{\Delta}$ by its αth and βth columns, we see that $\Delta = (-1)^{\alpha+\beta+1}\bar{\Delta}$. Now

$$y_\alpha(\alpha) = \det \begin{pmatrix} 1 & x_{\pi(0)1} & \cdots & x_{\pi(0),\beta-1} & x_{\pi(0),\beta+1} & \cdots & x_{\pi(0)d} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{\pi(d-2),1} & \cdots & x_{\pi(d-2),\beta-1} & x_{\pi(d-2),\beta+1} & \cdots & x_{\pi(d-2)d} \\ 0 & 0 \cdots 0 & 1 & 0 \cdots & \cdots & \cdots & 0 \\ & & (\alpha) & & & & \end{pmatrix}$$

and

$$y_\beta(\alpha) = \det \begin{pmatrix} 1 & x_{\pi(0)1} & \cdots & x_{\pi(0),\alpha-1} & x_{\pi(0),\alpha+1} & \cdots & x_{\pi(0)d} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{\pi(d-2)1} & \cdots & x_{\pi(d-2),\alpha-1} & x_{\pi(d-2),\alpha+1} & \cdots & x_{\pi(d-2)d} \\ 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \end{pmatrix};$$

hence expanding by the last row we get

$$y_\alpha(\alpha) = (-1)^{d+\alpha+1}\Delta, \qquad y_\beta(\alpha) = 0.$$

Similarly, we get

$$y_\alpha(\beta) = 0, \qquad y_\beta(\beta) = (-1)^{d+\beta}\Delta.$$

It follows that the orientation of the points $(0, 0)$, $(y_\alpha(\alpha), y_\beta(\alpha))$, $(y_\alpha(\beta), y_\beta(\beta))$ in the x_α, x_β-plane, with the coordinates taken in that order, is

$$\det \begin{pmatrix} 1 & 0 & 0 \\ 1 & (-1)^{d+\alpha+1}\Delta & 0 \\ 1 & 0 & (-1)^{d+\beta}\Delta \end{pmatrix} = (-1)^{\alpha+\beta+1} = \Delta\bar{\Delta}.$$

But since we are relabeling (x_α, x_β) either (u, v) or (v, u)—depending on whether Δ is $+1$ or -1 (resp.)—it follows that in the u, v-plane the orientation of the points O, $f(P_\alpha), f(P_\beta)$ is precisely $\bar{\Delta}$, i.e., matches that of the original $(d+1)$-tuple $P_0, \cdots, P_d$. A simple continuity argument then shows that the same conclusion must hold for *any* choice of $P_0, \cdots, P_d$ in general position. (The point is that our formulas for (y_α, y_β) give the correct projection to within a dilatation and/or a reflection in one or both coordinate axes, so they preserve collinearity and either preserve or reverse orientation; the above shows that they do, in fact, preserve it.)

Step 6 translates $f(P_{i_1})$ to the origin, and step 7 designates j as "good" if $f(P_j) \neq f(P_{i_1})$, i.e., if P_j is not in the flat $\langle P_{i_1}, \cdots, P_{i_{d-1}}\rangle$. In step 8, we compute the reflection in the origin of only the good points P_j with indices $j > i_{d-2}$, since these are all for which we have to compute $\lambda(i_1, \cdots, i_{d-2}, j)$; on the other hand we must take into account *all* the points P_k, even those with lower indices, when we count those on the positive side of $\langle P_{i_1}, \cdots, P_{i_{d-1}}, P_j\rangle$, hence we need *all* the slopes. Step 9 does the sorting for the good indices $j > i_{d-2}$, and steps 10 and 11 compute λ. In step 12, finally, we extend the λ-function to all ordered d-tuples; notice that every ordered d-tuple $(j_1, \cdots, j_d)$ with the j_k distinct and with $1 \leqq j_k \leqq n$ for every k is an even permutation of an ordered d-tuple $(i_1, \cdots, i_d)$, in fact of a unique one, with $i_1 < \cdots < i_{d-2} < \{i_{d-1}, i_d\}$—hence we never compute λ more than once for the same set of distinct indices. (If desired for the sake of compactness, in fact, the λ-array can be computed only for d-tuples with $i_1 < \cdots < i_{d-2} < \{i_{d-1}, i_d\}$; however, for comparison purposes this will not be enough, since when we renumber the points the indices will no longer form an increasing sequence—see § 5 below.)

The time required, for each $i_1, \cdots, i_{d-1}$ in step 3, is $O(d^2)$; in step 4 it is $O(d^2 n)$ (or faster), in steps 5, 6, 7 and 8 it is $O(n)$, in step 9 $O(n \log n)$, in steps 10 and 11 $O(n)$, and in step 12 $O(d!n)$. Since there are $O(n^{d-1}/(d-1)!)$ increasing $(d-1)$-tuples $i_1, \cdots, i_{d-1}$, the total time for Algorithm 3.1 is therefore

$$O((n^{d-1}/(d-1)!)(n(d!+\log n))) = O(n^d(d!+\log n)/(d-1)!).$$

In particular, if we think of d as fixed and n large, the time is $O(n^d \log n)$, which generalizes the time of $O(n^2 \log n)$ for Algorithm 2.1.

4. The canonical orderings of a configuration in $\mathbb{R}^d$. If a configuration $\mathscr{C}$ of points is given on a directed line, there is precisely one canonical way to order them; two if the line has no preferred orientation. But in the plane, or in higher dimensions, this is no longer true. Of course if our space is coordinatized, we can order lexicographically, say, by the coordinates; although such an ordering is useful for certain purposes, it is not intrinsic to the set, and would change if we performed an orientation-preserving affine transformation on the set; thus sets of the same "shape" or order type would have a great many such orderings. On the other hand, we could order the points of $\mathscr{C}$ by letting a hyperplane sweep across them; this would give, in general, $n(n-1)$ distinct orderings for a planar configuration, but two configurations of the same order type might have different sets of these orderings, even totally disjoint ones! (see [12, Thm. 1.7 (iv)$\Rightarrow$(vi)] for such an example). It is, however, possible to distinguish $k \leqq n$ of the $n!$ possible orderings of a planar configuration $\mathscr{C}$, where k is the number of vertices of the convex hull of $\mathscr{C}$, that turn out to be invariant under order-type-preserving transformations, and that we shall find useful in § 5 when we want to compare randomly numbered configurations. These we shall call the *canonical orderings* associated to $\mathscr{C}$. They are defined as follows:

If $\mathscr{C} = \{P_1(x_1, y_1), \cdots, P_n(x_n, y_n)\}$ is a configuration in the plane, and if P_i is any vertex of the convex hull conv $(\mathscr{C})$, consider a ray with origin P_i that begins by pointing

away from conv ($\mathscr{C}$), and that rotates counterclockwise. We list P_i first, then the remaining points $P_j (j \neq i)$ in the order in which the ray encounters them; if it meets $P_{j_1}, \cdots, P_{j_r}$ simultaneously, we list these in order of increasing distance from P_i. (If there are repeated points, we list these in any order at all.) This is illustrated in Fig. 2, with $i = 5$.

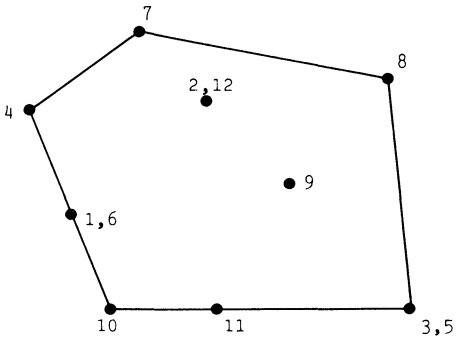


FIG. 2. 3, 5, 8, 9, 12, 2, 7, 4, 1, 6, 11, 10.

It is important to realize that if another configuration $\mathscr{C}' = \{P_1'(x_1', y_1'), \cdots, P_n'(x_n', y_n')\}$ is equivalent to $\mathscr{C}$, for example if $\mathscr{C}'$ is the image of $\mathscr{C}$ under an orientation-preserving affine transformation, then

(1) P_i will be an extreme point of $\mathscr{C}$ if and only if P_i' is an extreme point of $\mathscr{C}'$, and

(2) the canonical ordering of $\{1, \cdots, n\}$ associated to P_i will agree with that associated to P_i' (or will agree up to a permutation of the points within each class of repeated points if these happen to exist).

We can do the same thing in higher dimensions, as soon as we have the notion of a *face-flag*. If Π is a d-dimensional polytope lying in $\mathbb{R}^d$, we call a sequence $(\sigma_0, \cdots, \sigma_{d-2})$ of faces of Π a *face-flag*, if each σ_j is a face of σ_{j+1}, and if each σ_j is a face of Π of dimension j. (Note that if $\mathscr{C} = \{P_1, \cdots, P_n\}$ is a configuration in $\mathbb{R}^d$, and if $\Sigma = (\sigma_0, \cdots, \sigma_{d-2})$ is a face-flag of conv ($\mathscr{C}$), there are points $P_{i_0}, \cdots, P_{i_{d-2}} \in \mathscr{C}$ such that $\sigma_j = \langle P_{i_0}, \cdots, P_{i_j} \rangle$ for each $j = 0, \cdots, d-2$.) Now imagine a hyperplane H which "rotates around" σ_{d-2} in the positive direction (as determined by the coordinates in the order $x_1, \cdots, x_d$), starting in a supporting position. It meets the points of $\mathscr{C}$, perhaps several at a time, in some order $\mathscr{C}_1, \cdots, \mathscr{C}_r$, where all the points of σ_{d-2} are thought of as included in $\mathscr{C}_1$. Each subset $\mathscr{C}_i$ is a configuration lying in a $(d-1)$-dimensional space, hence the face-flag Σ determines an ordering of $\mathscr{C}_i$ into subsets $\mathscr{C}_{i1}, \cdots, \mathscr{C}_{is_i}$, by "rotating" a $(d-2)$-flat "around" σ_{d-3}, beginning with σ_{d-2} and moving through $\mathscr{C}_i$. We continue inductively, lowering the dimension by 1 each time. For the configuration illustrated in Fig. 3, for example, the ordering 3, 8, 5, 7, 2, 4, 1, 6 is induced by the face-flag $\langle 3 \rangle, \langle 3, 5 \rangle$.

There is another way to think of this canonical ordering, which we shall use in Algorithm 4.1 below. If $u_1, \cdots, u_d$ are Cartesian coordinates in $\mathbb{R}^d$, with origin O, we can define a spherical coordinate system $(\phi_1, \cdots, \phi_{d-1}, \rho)$ by letting, for each point $P \in \mathbb{R}^d$, $\phi_1(P) =$ the angle between $\overrightarrow{OP}$ and the positive u_1-axis, $\phi_2(P) =$ the angle between $\overrightarrow{OP^{(2)}}$ and the positive u_2-axis, where $P^{(2)}$ is the projection of P in the $u_2, \cdots, u_d$-hyperplane, $\cdots$, $\phi_{d-1}(P) =$ the angle between $\overrightarrow{OP^{(d-1)}}$ and the positive u_{d-1}-axis, where $P^{(d-1)}$ is the projection of P in the u_{d-1}, u_d-plane, and $\rho = |OP|$; in

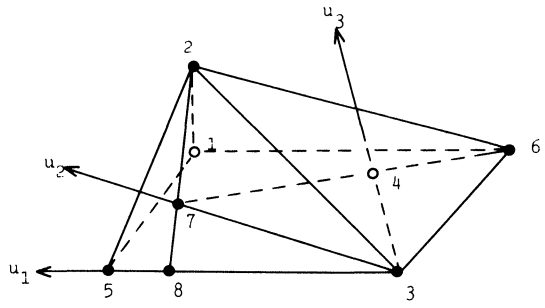


FIG. 3,

each case, if $\overrightarrow{OP}^{(i)} = 0$, call the corresponding angle 0 also. (This generalizes the usual spherical coordinate system in $\mathbb{R}^3$.) If $\mathscr{V} = \{\vec{v}_1, \cdots, \vec{v}_d\}$ is any real basis of the vector space $\mathbb{R}^d$, we may interpret the $\mathscr{V}$-coordinates $u_1, \cdots, u_d$ of any point P arising from the representation $\overrightarrow{OP} = \sum u_i \vec{v}_i$ as the Cartesian coordinates of a corresponding point P' in another copy of $\mathbb{R}^d$ (this amounts to performing an orientation-preserving linear transformation of $\mathbb{R}^d$ to itself), and compute the corresponding spherical coordinates $\phi_1, \cdots, \phi_{d-1}, \rho$. In particular, if $\sum = \sigma_0, \cdots, \sigma_{d-2}$ is a face-flag of conv $(\mathscr{C})$ and if $\sigma_j = \langle P_{i_0}, \cdots, P_{i_j} \rangle$ for every j, we may choose points $P_{i_{d-1}}$, P_{i_d} such that $P_{i_{d-1}}$ is on one of the two $(d-1)$-faces of conv $(\mathscr{C})$ meeting along σ_{d-2} and P_{i_d} is off $\langle P_{i_0}, \cdots, P_{i_{d-1}} \rangle$, and such that $P_{i_0}, \cdots, P_{i_d}$ has *positive* orientation (this determines *which* of the two faces $P_{i_{d-1}}$ is on!). Then take for $\mathscr{V}$ in the construction above the basis

$$\overrightarrow{P_{i_0}P_{i_1}}, \cdots, \overrightarrow{P_{i_0}P_{i_d}}.$$

This determines a coordinate system, hence (as above) a spherical coordinate system, and it is easy to see that the canonical ordering of $\{1, \cdots, n\}$ determined by the face-flag $\sigma_0, \cdots, \sigma_{d-2}$ is nothing more than the lexicographic ordering by $\phi_{d-1}, \cdots, \phi_1, \rho$. In Fig. 3, for example, the face-flag $\langle 3 \rangle$, $\langle 3, 5 \rangle$ can be augmented by the points 7, 4 (since 3, 5, 7, 4 has positive orientation in the "right-hand screw" sense), and this gives the basis $\overrightarrow{35}$, $\overrightarrow{37}$, $\overrightarrow{34}$, which—in turn—produces the ordering 3, 8, 5, 7, 2, 4, 1, 6 already indicated.

It is now easy to give a fast algorithm for determining the canonical ordering of a configuration $\mathscr{C}$ induced by a face-flag. The idea, as already suggested, is to compute the (generalized) spherical coordinates of each point of $\mathscr{C}$ with respect to the affine coordinate system induced by the face-flag, and to sort the points of $\mathscr{C}$ lexicographically with respect to their spherical coordinates. For ease of computation, we have chosen to use the squares of the cosines of the angles, rather than the angles themselves, as a sorting key (see Analysis below).

ALGORITHM 4.1.

Input: $(P_i(x_{i1}, \cdots, x_{id}))$, $i = 1, \cdots, n$, and $i_0, \cdots, i_{d-2} \in \{1, \cdots, n\}$, where $(\langle P_{i_0}, \cdots, P_{i_j} \rangle)$, $j = 0, \cdots, d-2$, is a face-flag of conv $(\{P_1, \cdots, P_n\})$.

Output: $\pi(1), \cdots, \pi(n)$, the permutation of $1, \cdots, n$ that is canonically associated to the given face-flag.

1. Step 3 of Algorithm 3.1 (with $i_0, \cdots, i_{d-2}$ in place of $i_1, \cdots, i_{d-1}$), yielding α, β, σ. [N.B.: The determinant in question cannot vanish for all j, k.]

2. Step 4 of Algorithm 3.1, yielding $y_\alpha(j)$, $y_\beta(j)$ for every $j = 1, \cdots, n$.

3. If $\sigma = +1$, let $u'(j) = y_\alpha(j)$ and $v'(j) = y_\beta(j)$ for each $j = 1, \cdots, n$; if $\sigma = -1$, let $u'(j) = y_\beta(j)$ and $v'(j) = y_\alpha(j)$ for each $j = 1, \cdots, n$.

4. For each $j = 1, \cdots, n$, let $u_j = u'(j) - u'(i_1)$ and $v_j = v'(j) - v'(i_1)$.

5. For each $j = 1, \cdots, n$, if $(u_j, v_j) \neq (0, 0)$, call j "good".

6. For every good $j = 1, \cdots, n$, let $u_{n+j} = -u_j$, $v_{n+j} = -v_j$, and let $m_j = m_{n+j} = v_j/u_j$ if $u_j \neq 0$.

7. Sort the indices $\{j \mid j \text{ is good}\} \cup \{n+j \mid j \text{ is good}\}$ into subsets, as follows:

(a) first those for which $u_j > 0$, using m_j as key;

(b) next those for which $u_j = 0$ and $v_j > 0$;

(c) next those for which $u_j < 0$, using m_j as key;

(d) finally those for which $u_j = 0$ and $v_j < 0$.

(In (a) and (c) we get a list of subsets; the order within each subset is irrelevant.) Say the result is

$$J_{11}, \cdots, J_{1s_1}, \cdots, J_{r1}, \cdots, J_{rs_r},$$

where the points with indices $J_{k1}, \cdots, J_{ks_k}$ constitute an entire subset, and there are r subsets all together.

8. For each $k = 1, \cdots, r$, let

$$n_k = |\{m \mid 1 \leqq m \leqq s_k, J_{km} \leqq n\}|.$$

9. If $n_1 = 0$, let k' be the first k with $n_k \neq 0$ and let $k'' = k' + r/2 - 1$; if $n_1 \neq 0$, let k'' be (the first k with $n_k = 0$) $- 1$, and let

$$k' = \begin{cases} k'' + r/2 + 1 & \text{if this is} \leqq r, \\ 1, & \text{otherwise.} \end{cases}$$

10. Let $i_{d-1} = j_{k'1}$, $i_d = j_{k''1}$.

11. For every $j = 1, \cdots, n$ and every $k = 1, \cdots, d$, let $z_{jk} = x_{jk} - x_{i_0 k}$.

12. For every $p = 1, \cdots, d$ and every $q = 1, \cdots, d$, let

$$A_{pq} = (-1)^{p+q} \det ((z_{i_k m})_{k \neq q, m \neq p}).$$

13. For every $j = 1, \cdots, n$, and every $q = 1, \cdots, d$, let

$$w_{jq} = \sum_{p=1}^{d} z_{jp} A_{pq}.$$

14. For every $j = 1, \cdots, n$ and every $q = 1, \cdots, d$, let

$$\Phi_{jq} = \begin{cases} \dfrac{(-\operatorname{sgn} w_{jq}) w_{jq}^2}{w_{jq}^2 + \cdots + w_{jd}^2} & \text{if the denominator} \neq 0, \\ 1, & \text{otherwise.} \end{cases}$$

15. For every $j = 1, \cdots, n$, let $R_j = w_{j1}^2 + \cdots + w_{jd}^2$.

16. Sort $j = 1, \cdots, n$ lexicographically by $\Phi_{d-1}, \cdots, \Phi_1, R$, to determine the permutation π such that $\pi(1) \leqq \cdots \leqq \pi(n)$ in this lexicographic order.

17. Stop.

Analysis. Steps 1 through 8, just as in Algorithm 3.1, determine the number of points in each ray when the configuration is projected along the flat $\langle P_{i_0}, \cdots, P_{i_{d-2}} \rangle$ into a coordinate plane meeting the flat transversally. Since we know in advance that $\langle P_{i_0}, \cdots, P_{i_{d-2}} \rangle$ is an edge-flat, it follows that the projected configuration has $f(P_{i_0})$ as an extreme point, where f is the projection. Steps 9 and 10 locate two vertices projecting onto the edges adjacent to that extreme point, and chooses them so that the orientation of $f(P_{i_0})$, $f(P_{i_{d-1}})$, $f(P_{i_d})$ will be positive, hence also the orientation of

$P_{i_0}, \cdots, P_{i_{d-2}}, P_{i_{d-1}}, P_{i_d}$. Step 11 translates P_{i_0} to the origin. Step 12 computes the adjoint A of the matrix of components of the basis vectors $\overrightarrow{P_{i_0}P_{i_1}}, \cdots, \overrightarrow{P_{i_0}P_{i_d}}$, so that in step 13 we can get the components $w_1, \cdots, w_d$ of each vector $\overrightarrow{P_{i_0}P_j}$ in terms of this basis, using the standard change-of-basis formula from linear algebra. If ϕ_q is the qth spherical coordinate, $1 \leqq q \leqq d-2$, as defined in the discussion preceding Algorithm 4.1, its cosine is given by

$$\cos \phi_q = \begin{cases} w_q(w_q^2 + \cdots + w_d^2)^{-1/2} & \text{if } w_q^2 + \cdots + w_d^2 \neq 0, \\ 1, & \text{otherwise.} \end{cases}$$

To avoid unnecessary computations, we can use the squares of these numbers, together with the appropriate signs, as a sorting key, rather than the numbers themselves. In a similar way, we can use $R = \rho^2$ instead of ρ. Since we want to sort in order of increasing ϕ, i.e. decreasing $\cos \phi$, the signs have been reversed in the definition of Φ_q in step 14, so that—in step 16—we can sort with respect to the increasing order in *all* the variables.

The time for steps 1 through 8, as for steps 3 through 10 of Algorithm 3.1 for a single set of indices, is $O(d^3 + n \log n)$, and steps 9 and 10 are shorter. Step 11 takes $O(dn)$, step 12 takes $O(d^5)$, and steps 13, 14 and 15 take $O(d^2 n)$. Step 16 takes $O(dn \log n)$, giving a total time of $O(d^5 + d^2 n + dn \log n)$ for Algorithm 4.1, i.e. $O(n \log n)$ for d fixed.

5. An algorithm for comparing randomly numbered configurations. Except for the way in which two sets of points on a line have been labeled, they "look" the same from the point of view of their order. This is not true in the plane, however, or in higher dimensions. There is no way to permute the labels on the sets in Fig. 4a and b to get their λ- or Λ-functions to match up. Thus, for *unlabeled* sets of n points in dimension $\geqq 2$, many essentially distinct order types are possible.

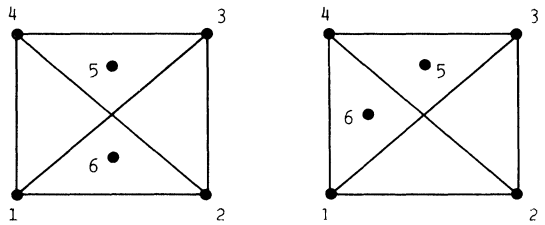
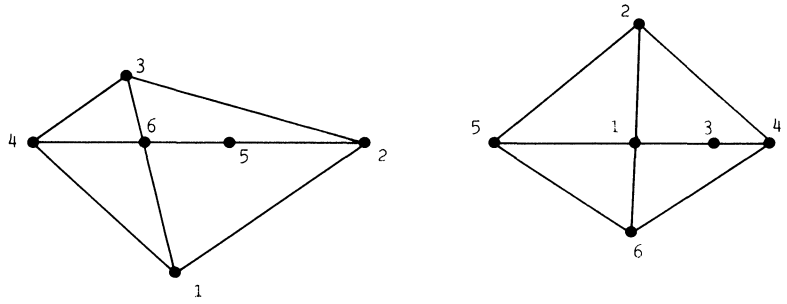


FIG. 4.

If $\mathscr{C}$ and $\mathscr{C}'$ are numbered configurations in $\mathbb{R}^d$, and we wish to compare them to see whether—*with the given numberings*—they are equivalent, we need only compare $\lambda(i_1, \cdots, i_d)$ for every $i_1, \cdots, i_d$. But suppose the given numberings do not match; how can we nevertheless determine whether they have the same order type, i.e., whether their points can be put in $1-1$ correspondence (and in fact find *all* such matchings) so that corresponding $(d+1)$-tuples are similarly oriented? If they are given in the form $\mathscr{C} = \{P_1, \cdots, P_n\}$, $\mathscr{C}' = \{P_1', \cdots, P_n'\}$, and we have encoded their (numbered) order types by calculating the λ-function of each, it would seem that we would have to try all the $n!$ possible renumberings of $\mathscr{C}'$ to be sure of finding all those (if any) in which the λ-function would agree with that of $\mathscr{C}$. But it turns out that we can eliminate most of these renumberings from consideration before we start. Let us



(a) $C=\{P_1,\cdots,P_6\}$ (b) $C'=\{P'_1,\cdots,P'_6\}$

FIG. 5.

illustrate this first by an example in the plane, where we can reduce the number of necessary renumberings to n, at most.

Fig. 5 shows two configurations that clearly have the same order type, but not in the numbering shown. How can we recognize this from their λ-matrices? We have

$$(\lambda_{ij})=\begin{pmatrix} \omega & 4 & 1 & 0 & 3 & 1\\ 0 & \omega & 4 & 1 & 1 & 1\\ 2 & 0 & \omega & 4 & 1 & 2\\ 4 & 1 & 0 & \omega & 1 & 1\\ 1 & 1 & 3 & 1 & \omega & 1\\ 2 & 1 & 1 & 1 & 1 & \omega \end{pmatrix}, \qquad (\lambda'_{ij})=\begin{pmatrix} \omega & 1 & 1 & 1 & 1 & 2\\ 2 & \omega & 1 & 0 & 4 & 2\\ 1 & 3 & \omega & 1 & 1 & 1\\ 1 & 4 & 1 & \omega & 1 & 0\\ 1 & 0 & 1 & 1 & \omega & 4\\ 1 & 1 & 3 & 4 & 0 & \omega \end{pmatrix},$$

and the problem is to find all permutations π of $\{1,\cdots,n\}$ such that $\lambda'_{\pi(i)\pi(j)}=\lambda_{ij}$ for every i, j. Consider the point P_1. Since it is a vertex of $\mathscr{C}$, $P'_{\pi(1)}$ will have to be a vertex of $\mathscr{C}'$ as well. *And the canonical ordering induced by $P'_{\pi(1)}$ in $\mathscr{C}'$ must agree (after the renumbering π) with that induced by P_1 in $\mathscr{C}$, namely* 125634. This means that if we list all of the vertices of $\mathscr{C}'$, and write down—for each—the canonical ordering it induces in $\mathscr{C}'$, π will have to map 125634 to one of those canonical orderings. Here we have 2, 4, 5 and 6 as vertices, inducing the following canonical orderings of $\mathscr{C}'$:

2:251634 5:561342

4:423156 6:643125.

Hence we need only try the four permutations

$$\begin{pmatrix}1&2&5&6&3&4\\2&5&1&6&3&4\end{pmatrix},\ \begin{pmatrix}1&2&5&6&3&4\\4&2&3&1&5&6\end{pmatrix},\ \begin{pmatrix}1&2&5&6&3&4\\5&6&1&3&4&2\end{pmatrix},\ \begin{pmatrix}1&2&5&6&3&4\\6&4&3&1&2&5\end{pmatrix}$$

to find π. Applying each of these in turn to λ gives

$$\begin{pmatrix} \omega & 1 & 3 & 1 & 1 & 1\\ 3 & \omega & 1 & 0 & 4 & 1\\ 1 & 2 & \omega & 4 & 0 & 2\\ 1 & 4 & 0 & \omega & 1 & 1\\ 1 & 0 & 4 & 1 & \omega & 1\\ 1 & 2 & 1 & 1 & 1 & \omega \end{pmatrix}, \begin{pmatrix} \omega & 1 & 1 & 2 & 1 & 1\\ 1 & \omega & 1 & 0 & 4 & 1\\ 1 & 1 & \omega & 1 & 3 & 1\\ 1 & 4 & 3 & \omega & 1 & 0\\ 2 & 0 & 1 & 2 & \omega & 4\\ 1 & 1 & 1 & 4 & 0 & \omega \end{pmatrix}, \begin{pmatrix} \omega & 1 & 1 & 3 & 1 & 1\\ 1 & \omega & 1 & 0 & 4 & 1\\ 1 & 1 & \omega & 1 & 2 & 1\\ 1 & 4 & 2 & \omega & 2 & 0\\ 3 & 0 & 1 & 1 & \omega & 4\\ 1 & 1 & 1 & 4 & 0 & \omega \end{pmatrix}, \begin{pmatrix} \omega & 1 & 1 & 1 & 1 & 2\\ 2 & \omega & 1 & 0 & 4 & 2\\ 1 & 3 & \omega & 1 & 1 & 1\\ 1 & 4 & 1 & \omega & 1 & 0\\ 1 & 0 & 1 & 1 & \omega & 4\\ 1 & 1 & 3 & 4 & 0 & \omega \end{pmatrix},$$

and we see that the last of these agrees with λ', so we must have

$$\pi = \begin{pmatrix} 1 & 2 & 5 & 6 & 3 & 4 \\ 6 & 4 & 3 & 1 & 2 & 5 \end{pmatrix}.$$

This idea extends to higher dimensions. There we must consider not only all the vertices of conv $(\mathscr{C})$, but all of its face-flags, as defined in § 4. Each of these induces a canonical ordering, and two configurations $\mathscr{C}$ and $\mathscr{C}'$ will have the same order type under the correspondence induced by $\pi : \{1, \cdots, n\} \to \{1, \cdots, n\}$ only if a fixed canonical ordering of $\{1, \cdots, n\}$ induced by the face-flag Σ (say) is transformed, by π, into the canonical ordering induced by the image $\Sigma^{(\pi)}$ of Σ in $\mathscr{C}'$. It is therefore sufficient to try all the permutations $\pi \in S_n$ which arise by associating to a fixed canonical numbering of $\mathscr{C}$ all the canonical numberings of $\mathscr{C}'$ (let us call such a renumbering of $\mathscr{C}$, induced by π, $\mathscr{C}^{(\pi)}$); among these will be all of the matching numberings, if any, of $\mathscr{C}$ and $\mathscr{C}'$. (If $\mathscr{C}$ has repeated points, then any renumbering of $\mathscr{C}$ which merely permutes the points within the groups of equal ones has no effect on its λ-function; for this reason it is sufficient, when obtaining the canonical numbering of a configuration associated to a face-flag, to list repeated points in any arbitrary order.)

How many face-flags are there for a configuration of n points in $\mathbb{R}^d$, and how do we find them all? It is not difficult to give a sharp upper bound on the number $\mathscr{F}(d, n)$ of face flags, hence of canonical orderings:

$$\mathscr{F}(d, n) \leqq \frac{d!}{2}\left[\binom{n - \lfloor (d+1)/2 \rfloor}{n-d} + \binom{n - \lfloor (d+2)/2 \rfloor}{n-d}\right].$$

This follows from the fact that there are no more than

$$\binom{n - \lfloor (d+1)/2 \rfloor}{n-d} + \binom{n - \lfloor (d+2)/2 \rfloor}{n-d}$$

faces of dimension $d-1$ for a simplicial polytope of dimension d, with equality for cyclic polytopes [15], and each such face gives rise to $d!$ face-flags $\sigma_0 \subset \sigma_1 \subset \cdots \subset \sigma_{d-1}$, of which precisely half have positive orientation; if a polytope is not simplicial, triangulating it will only increase the number of faces in each dimension, and it is easy to describe an injective map from the set of face-flags of the original polytope to the set of face-flags of its triangulaion. Hence we have

PROPOSITION 5.1. *A configuration of n points in the plane has $\leqq n$ canonical orderings, in 3-space has $\leqq 6n - 12$, and in $\mathbb{R}^d$ for d fixed has $O(n^{\lfloor d/2 \rfloor})$.*

As to the question of how to find all the face-flags, we shall describe four methods that can be used, depending on the circumstances:

Remark 5.2. (a) If $\mathscr{C}$ is a configuration in the plane, its face-flags are just its vertices, hence can be found by any algorithm which locates the vertices of the convex hull of a planar configuration [20]. They can also be found easily during the calculation of the λ-matrix, as in Algorithm 2.1, by inserting the following additional step after step 7:

7.1. If $\lambda(i, j) = 0$ and $n_{k(j)} = s_{k(j)}$, P_i is a vertex.

In fact, we can save time by calculating the canonical numberings themselves, during the sorting, as follows:

(i) Test each point P_i to see if it is an extreme point by checking whether $\lambda(i, j) = 0$ for some j and—if so—whether $n_{k(j)} = s_{k(j)}$ (i.e., whether the ray containing P_j contains no reflected points; this device is used again in method (b) below).

(ii) If i and j satisfy these conditions, sort the points $P_{J_{k1}}, \cdots, P_{J_{ks_k}}$ within the kth ray, for each k, by their distances from the origin (i.e., from P_i); let the result be $\bar{J}_{k1}, \cdots, \bar{J}_{ks_k}$.

(iii) Record the ordering

$$i_1, \cdots, i_p, \bar{J}_{k(j)+1,1}, \cdots, \bar{J}_{r,s}, \bar{J}_{11}, \cdots, \bar{J}_{k(j),s_{k(j)}},$$

where $i_1, \cdots, i_p$ are the "bad" indices, i.e., $P_{i_1} = \cdots = P_{i_p} = P_i$ (i will, of course, be among them). This will be the canonical ordering associated to the extreme point P_i.

(b) If $\mathscr{C}$ is a configuration in $\mathbb{R}^3$, its face-flags are just its directed edges, and there are algorithms (for example [3]) which locate all the edges of the convex hull of a 3-dimensional configuration. They can also be found during the calculation of the λ-function, by inserting the following additional steps after step 11 in Algorithm 3.1:

11.1. If $\lambda(i_1, i_2, j) \neq 0$ for all j, go to step 12; otherwise, let j_0 be the first j for which $\lambda(i_1, i_2, j) \neq 0$.

11.2. If $n_{k(j_0)} \neq s_{k(j_0)}$, go to step 12.

11.3. If $\gamma = \{1, 2, 3\} \backslash \{\alpha, \beta\}$, find $j_{\min}$ and $j_{\max}$, where

$$x_\gamma(j_{\min}) = \min \{x_\gamma(j) \mid j \text{ is not "good"}\}, \quad \text{and}$$

$$x_\gamma(j_{\max}) = \max \{x_\gamma(j) \mid j \text{ is not "good"}\}.$$

11.4. Enter the pairs $(j_{\min}, j_{\max})$ and $(j_{\max}, j_{\min})$ in the list of directed edges.

In addition, since we need the reflections of *all* the points in O in order to decide whether O is a vertex, Algorithm 3.1 should be further modified by executing steps 8, 9 and 11 for *every* good j, not merely those $\geqq i_1 + 1$; therefore we need only execute step 12 for the even permutations π that *fix the last index*.

(c) If $\mathscr{C}$ is a configuration in $\mathbb{R}^d$, for any $d \geqq 2$, and it is known that the points of $\mathscr{C}$ are in general position, i.e. that every set of $k+1 \leqq d$ points of $\mathscr{C}$ spans a k-flat,[1] then after sorting $\mathscr{C}$ we can get the face-flags of $\mathscr{C}$ very simply from the λ-function, as follows:

Whenever $\lambda(i_0, \cdots, i_{d-1}) = n - d$, then $i_0, \cdots, i_{d-2}$ determines a face-flag of $\mathscr{C}$.

If we let i_d be an arbitrary index $\neq i_0, \cdots, i_{d-1}$, then this gives, in fact, the *full* positively oriented face flags

$$\{P_{i_0}\} \subset \{P_{i_0}, P_{i_1}\} \subset \cdots \subset \{P_{i_0}, \cdots, P_{i_d}\},$$

so that we can omit steps 1 through 10 of Algorithm 4.1 if we wish to calculate the canonical orderings of $\mathscr{C}$.

Notice that if n points are given at random in $\mathbb{R}^d$, they *will* be in general position, in the sense above, since the set of n-tuples of points in $\mathbb{R}^d$ not in general position has measure zero in any bounded region of positive content. But all that is really needed to use method (c) is the weaker condition that conv ($\mathscr{C}$) be a *simplicial* polytope, i.e., that all of its $(d-1)$-faces be simplices. And it is very simple to check this from the λ-function: just check that whenever $\lambda(i_0, i_1, i_2, \cdots, i_{d-1}) = 0$ then $\lambda(i_1, i_0, i_2, \cdots, i_{d-1}) = n - d$. Thus method (c) will be applicable in many situations where the points of $\mathscr{C}$ are generated by some process with a random component.

[1] This can of course be checked from the λ-function itself: an obvious necessary and sufficient condition is that $\lambda(i_0, i_1, i_2, \cdots, i_{d-1}) + \lambda(i_1, i_0, i_2, \cdots, i_{d-1}) = n - d$ for every $i_0, \cdots, i_{d-1}$.

(d) If conv ($\mathscr{C}$) is not simplicial, or is not known to be, we cannot use method (c) to find its face-flags. If, in addition, the dimension of $\mathscr{C}$ is greater than 3, it then becomes necessary to resort to an algorithm, such as [3], that finds the $(d-1)$-faces of conv ($\mathscr{C}$), and to extend this recursively to find the complete lattice of faces of conv ($\mathscr{C}$). This will of course be rather time-consuming if d is large; it is, however, necessary if all the canonical orderings of $\mathscr{C}$ are desired, since these are in 1–1 correspondence with the face-flags, as we have seen, and the complete knowledge of these, in turn, is equivalent to the knowledge of the face lattice of conv ($\mathscr{C}$). Alternatively, we could use Algorithm 5.6 below, with step 2 modified by replacing the words "the first" by "each (in turn)"; this would give a time of $O(n^{d(d+3)/2})$ for finding all the face-flags, which is undoubtedly inefficient, albeit still polynomial.

We can now give the algorithm for sorting and comparing two randomly numbered configurations:

ALGORITHM 5.3.

Input: $\mathscr{C} = \{P_1(x_{11}, \cdots, x_{1d}), \cdots, P_n(x_{n1}, \cdots, x_{nd})\}$ and

$$\mathscr{C}' = \{P_1'(x_{11}', \cdots, x_{1d}'), \cdots, P_n'(x_{n1}', \cdots, x_{nd}')\}.$$

Output: $\{\pi \in S_n \mid \mathscr{C}^{(\pi)} \sim \mathscr{C}'\}$.

1. Sort $\mathscr{C}$ and $\mathscr{C}'$, using Algorithm 2.1 (if $d = 2$) or Algorithm 3.1 (if $d > 2$).
2. Using whichever of methods (a), (b), (c), (d) of Remark 5.2 applies, generate one face-flag $\Sigma = \langle i_0\rangle, \langle i_0, i_1\rangle, \cdots, \langle i_0, \cdots, i_{d-2}\rangle$ of $\mathscr{C}$.
3. Determine the permutation $\pi \in S_n$ corresponding to Σ, using Algorithm 4.1.
4. Similarly, generate *each* face-flag Σ' of $\mathscr{C}'$, in turn.
5. Determine the permutation $\pi' \in S_n$ corresponding to Σ', using Algorithm 4.1.
6. Let $\pi'' = \pi^{-1}\pi'$.
7. Compare $\lambda_{\mathscr{C}}(i_1, \cdots, i_d)$ and $\lambda_{\mathscr{C}'}(\pi''(i_1), \cdots, \pi''(i_d))$ for every choice of $i_1, \cdots, i_d$ with $i_1 < \cdots < i_{d-2} < \{i_{d-1}, i_d\}$. If they agree, record π''.
8. Return to step 4.
9. Stop.

Analysis. We have already seen what each of these steps does. As for the time required, it is quite variable. The sorting time (step 1), as we have seen, is $O(n^d \log n)$ for d fixed. Steps 2 and 4 can take anywhere from no appreciable extra time (if methods (a) or (b) apply) to an (indeterminately) long time if method (d) is invoked (but see Remark 5.5 below). In the case of a configuration in general position in $\mathbb{R}^d$, however (method (c)), we can always execute steps 2 and 4 in time $O(n^d)$. For each of the $O(n^{\lfloor d/2\rfloor})$ face-flags found in steps 2 and 4, steps 3 and 5 take time $O(n \log n)$ and step 7 takes $O(n^d)$; hence steps 3, 5, and 7—executed for all the face-flags—take $O(n^{\lfloor 3d/2\rfloor})$. This gives a total sorting and comparison time in the plane of $O(n^3)$, in $\mathbb{R}^3$ of $O(n^4)$, and in $\mathbb{R}^d (d \geqq 4)$—with a configuration in general position, or at least having a simplicial convex hull—of $O(n^{\lfloor 3d/2\rfloor})$.

Remark 5.4. This algorithm can also be used to find all the "order-symmetries" of a single configuration $\mathscr{C} = \{P_1, \cdots, P_n\}$ in $\mathbb{R}^d$, i.e., all permutations π of the indices giving equivalent configurations: just take $\mathscr{C}' = \mathscr{C}$.

Remark 5.5. When a configuration is sorted, if it is to be compared to other configurations, its canonical orderings should be determined at the same time. Then, when such a comparison is to be effected, only steps 6 and 7 of Algorithm 5.3 need be executed for each canonical ordering π' of $\mathscr{C}'$; this will cut down the comparison time considerably. This observation is particularly useful in applications where a dictionary of "standard" order types is to be encoded, and new configurations are to

be "looked up" in the dictionary (see for example Remark 6.4(a) below). In this case, when each "standard" configuration $\mathscr{C}'$ is encoded, we can compute all of its canonical orderings once and for all, and then whenever a new configuration $\mathscr{C}$ is offered for comparison it is only necessary to find *one* of its canonical orderings. Thus, after preprocessing the standard configurations, each new one will take only its sorting time $O(n^d \log n)$ to process, since *one* face-flag can easily be found in less time than that, no matter which of methods (a), (b), (c), or (d), is used.

Here, therefore, is an algorithm for finding *one* face-flag of a configuration in dimension $d \geqq 2$, directly from the λ-function of $\mathscr{C}$; it can be used when method (d) is needed:

ALGORITHM 5.6.

Input: The λ-function $(\lambda(i_1, \cdots, i_d))$, $1 \leqq i_j \leqq n$, $1 \leqq j \leqq d$, of a configuration $\mathscr{C}$.

Output: $j_0, \cdots, j_d$ such that $\langle j_0 \rangle, \cdots, \langle j_0, \cdots, j_d \rangle$ is a (positively oriented) face-flag of $\mathscr{C}$.

1. Let $D = d$ and let $S = \{1, \cdots, n\}$.
2. Find the first $i_1, \cdots, i_D \in S$ such that $\lambda(i_1, \cdots, i_D) = 0$.
3. Let $S = F$ and let $F = \{k \in S \mid \lambda(k, i_2, \cdots, i_D) = 0$ or $\lambda(i_1, k, i_3, \cdots, i_D) = 0$ or $\cdots$ or $\lambda(i_1, \cdots, i_{D-1}, k) = 0\}$.
4. Let j_D be the first index of S not in F.
5. For every $k_1, \cdots, k_{D-1} \in F$, let $\lambda_F(k_1, \cdots, k_{D-1}) =$ the number of indices $k \in F$ such that $\lambda(k_1, \cdots, k_{D-1}, k) = 0$; if $\lambda(k_1, \cdots, k_{D-1}, k) = \omega$ for *every* $k \in F$, let $\lambda_F(k_1, \cdots, k_{D-1}) = \omega$.
6. Replace D by $D - 1$, and λ by λ_F.
7. If $D > 0$, go to step 2.
8. Let $j_0 =$ the first index in F.
9. Stop.

Analysis. Step 2 gives us points $P_1, \cdots, P_d$ spanning a face of conv $(\mathscr{C})$ that has no points of $\mathscr{C}$ on its positive side. Step 3 gives us the complete set of points lying on that face, i.e.,

$$F = \{k \mid P_k \in \langle P_{i_1}, \cdots, P_{i_d} \rangle\};$$

this follows from Lemmas 1.5 and 1.6. Step 4 will give us the face-flag: first we find a point off face $\langle P_{i_1}, \cdots, P_{i_d} \rangle$, next a point off one of *its* faces, and so on, down to dimension 0. Step 5 determines the λ-function for the face F, so that we can proceed inductively.

For each value of D from d down to 1, the time required for step 2 is $O(n^D)$, for step 3 $O(Dn)$, for step 4 $O(n)$, for step 5 $O(n^D)$, giving $O(n^D)$ all together. Hence the total time for Algorithm 5.6 with d fixed is $O(n^d)$.

6. Further remarks, applications, and open problems. How many different order types of numbered configurations of n points are there in $\mathbb{R}^d$? This is a difficult question, but we can easily find an upper bound, and we can find evidence to conjecture a lower bound. On the one hand, the fact that every such configuration can be geometrically sorted, i.e., encoded by an $n \times n \times \cdots \times n$ d-array of integers $\leqq n$, implies—for information-theoretic reasons—that we cannot have more than $\exp(cn^d \log n)$ distinct order types. On the other hand, we venture

CONJECTURE 6.1. *The number of inequivalent configurations of n points in* $\mathbb{R}^d$ *is at least* $\exp(cn^d)$.

Let us adduce some evidence for this conjecture, at least in the planar case.

Just as the λ-matrix can be used to encode the order type of a configuration of points in $\mathbb{R}^2$, it can be used, more generally, to encode the order type of what is called a *generalized configuration* [4], [11], [12]. This consists of n points in $\mathbb{R}^2$ which have been joined pairwise by *pseudolines* L_{ij} forming an *arrangement*, i.e., by simple curves that are straight lines outside of a bounded region, and any two of which meet just once (and necessarily cross there). The order type of such a generalized configuration can be described completely in terms of the connecting pseudolines L_{ij} [12]: the condition that P_i, P_j, P_k have positive orientation amounts to saying that the connecting pseudolines must have directions occurring in the cyclic order $L_{ij}, L_{ik}, L_{jk}, L_{ji}, L_{ki}, L_{kj}$ (see Fig. 6). (Since each L_{ij} is eventually straight we can speak unambiguously of its

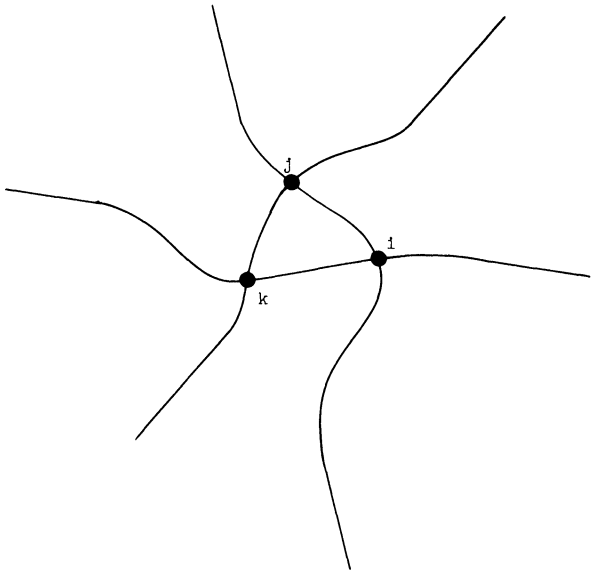


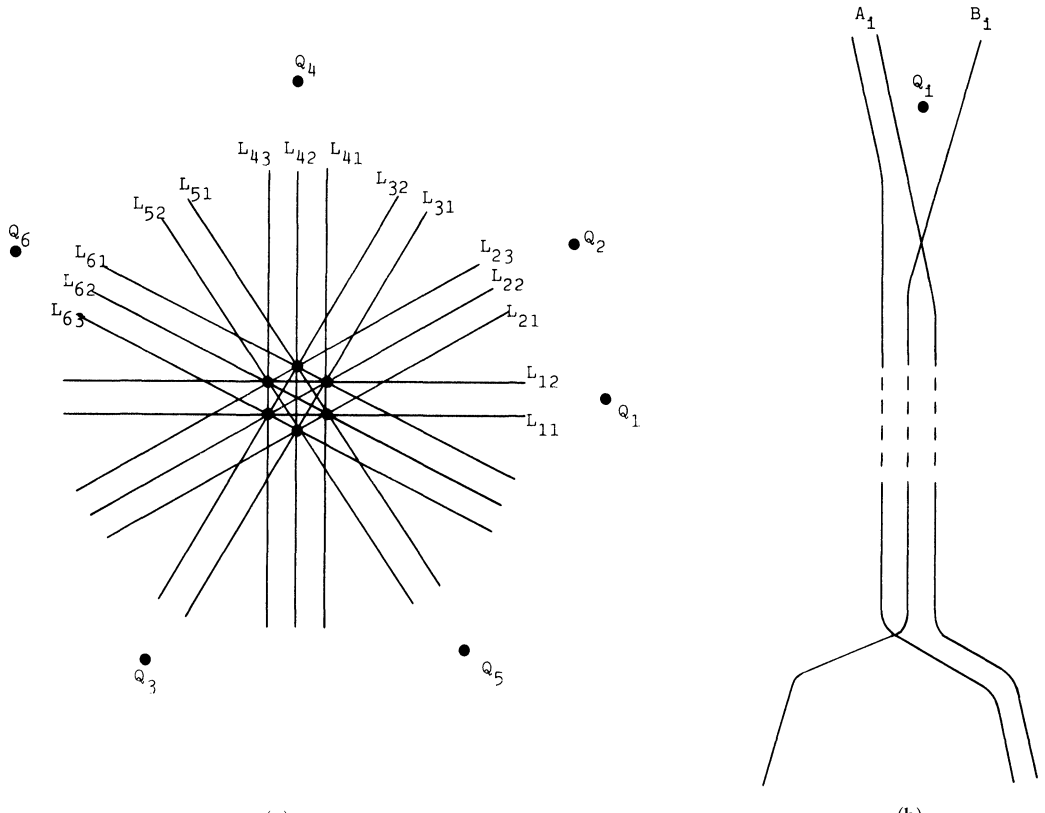
FIG. 6.

direction, which we take to be opposite to that of L_{ji}.) It follows from the main result of [10] that every generalized configuration of 8 or fewer points has the same order type as an ordinary configuration, while there are examples [9], [16] of 9 or more points for which this is not the case.

Theorem 1.8, which is the basis of our sorting procedure, holds for generalized configurations as well as ordinary ones [12], hence the order type of a generalized configuration is completely determined by its λ-matrix. While we cannot now prove Conjecture 6.1, even in the planar case, we do have

PROPOSITION 6.2. *There are at least* $2^{n^2/8}$ *generalized configurations of n points in the plane, for every n.*

Proof. Consider a regular k-gon with vertices $P_1, \cdots, P_k$, and draw all of its sides and diagonals extended fully (Fig. 7a). Let these be $L_{11}, \cdots, L_{1m_1}, \cdots, L_{k1}, \cdots L_{km_k}$, where $L_{i1}, \cdots, L_{im_i}$ form a complete set of parallels for each i. (If k is odd, we have $m_1 = \cdots = m_k = (k-1)/2$, while if k is even we have $m_1 = k/2, m_2 = k/2-1, m_3 = k/2, m_4 = k/2-1$, etc.) Now k new points, $Q_1, \cdots, Q_k$, are to be added, and we specify arbitrarily, for each i, which of $L_{i1}, \cdots, L_{im_i}$ are to come before Q_i, say L_{ij} for $j \in B_i$, and which after, say L_{ij} for $j \in A_i$, in the counterclockwise sense. In order to insert



(a) (b)

FIG. 7.

the Q_i, we first bend each L_{ij} with $j \in B_i$ slightly in the clockwise direction, and each L_{ij} with $j \in A_i$ slightly in the counterclockwise direction, and then insert Q_i after the two sets of L_{ij}'s have completely separated from each other (see Fig. 7b). Of course we must bend the other side of each L_{ij} as well, to ensure that the new L_{ij}'s, call them L_{ij}, still form an arrangement of pseudolines as defined above. Finally, we draw all the remaining connecting pseudolines P_iQ_j and Q_iQ_j, one at a time, with the help of the Levi enlargement lemma [16], which says that a pseudoline passing through two previously unconnected points can always be added to an arrangement to produce a new arrangement.

Since B_i was freely chosen for each i, and since two distinct choices of B_i for any i clearly give inequivalent configurations, regardless of the choices of the remaining B_j's, we have produced—for each even k—

$$(2^{k/2})^{k/2}(2^{k/2-1})^{k/2} = 2^{k^2/2-k/2},$$

and—for each odd k—

$$(2^{(k-1)/2})^k = 2^{k^2/2-k/2}$$

inequivalent generalized configurations, each having $2k$ points. Thus if $n = 2k$, we see that there are

$$2^{n^2/8-n/4}$$

inequivalent generalized configurations, each having n points, and if $n = 2k+1$ there are

$$2^{n^2/8-n/2+3/8}$$

such configurations. In each case, if we take into account the fact that each of the k new points can (independently) be placed on either side of the original k-gon, giving 2^k inequivalent versions of each generalized configuration, we can wipe out the linear terms in the exponent, and the result is that, as asserted, there are at least $2^{n^2/8}$ generalized configurations for every n.

Remark 6.3. By modifying the argument given, it is possible to improve the constant $\frac{1}{8}$ in Proposition 6.2 somewhat. If Conjecture 6.1 does turn out to be true, it will be an interesting problem to determine the constant c.

Remark 6.4. *Applications.* Since, as a result of Theorem 1.8 and Algorithm 5.3, it becomes possible to use a computer to sort and compare configurations of n points in $\mathbb{R}^d$ for moderate-sized n, a number of possible applications immediately suggest themselves. Among them are:

(a) *To pattern recognition.* There are various methods [1] of reducing an image to a point-pattern. Suppose we have a "dictionary" of standard images, each reduced to a black-and-white point pattern which has been encoded by its λ-matrix, and we wish to "look up" a new image, also encoded by its λ-matrix. If the transformation that has produced the new image is an affine orientation-preserving one, or at least one that preserves the relative orientations of points, such as a view of a solid object from a slightly different perspective than the "standard" view, the λ-matrices will agree, and they can be compared directly. If, on the other hand, the image has—in addition—undergone some local perturbations, as for example in hand-printed character analysis, its λ-matrix will not agree completely with a standard one, but will *correlate highly* with it. Thus one can measure the correlation of the λ-matrix with each one in the dictionary, and select the one giving the highest correlation; alternatively, if there are too many points to make such a comparison feasible, we can abstract *properties* of the standard λ-matrices (such as the proportion of extreme points, the number of hyperplanes cutting the configuration in half, and so on), and check off the corresponding properties of our new image against them. This procedure, especially when used in conjunction with other existing techniques (edge-detection, segmentation, noise-reduction), should prove highly useful in many scene- and pattern-analysis problems. For further details see [13].

(b) *To stereochemistry.* In [6] Dreiding and Wirth have suggested a method of encoding the order type of a configuration of points in $\mathbb{R}^d$, which they call a "multiplex", in order to provide an efficient way of distinguishing among stereoisomers, these being chemical compounds in which the same numbers of atoms are joined but with different orientations. (Each group of atoms that can exist in both a left- and a right-handed form is called a chirality element, and a single molecule may contain a number of these chemically distinguishable chirality elements.) Essentially, their encoding scheme consists of listing all the ordered $(d+1)$-tuples $i_1, \cdots, i_{d+1}$ with $i_1 < \cdots < i_{d+1}$ in lexicographic order, and writing 1 if the corresponding points are positively oriented and 0 if negatively. This gives a binary number for each order type, which they call its "signature", and amounts, essentially, to encoding what we called Λ earlier. (They consider only configurations in general position, and suggest a triadic representation if this is not the case.) Hence the storage (and therefore the minimum calculation time) for the signature of a configuration of n points in $\mathbb{R}^d$ is

$$\binom{n}{d+1} = \Omega(n^{d+1}) \quad \text{for } d \text{ fixed.}$$

Thus our sorting technique which consists of finding λ instead of Λ, whose time and storage requirement is only $O(n^d \log n)$, constitutes an improvement by a factor of $cn/\log n$, for n large, over the signature method. If, in applications to stereochemistry, one is interested in the orientations of only certain specified subsets (the chirality elements), a modification of our sorting scheme could easily be implemented, in which only each of these subsets is sorted; the result would then constitute an efficient means of encoding the various stereoisomers within a single class, which would facilitate their description and computer-aided identification.

(c) *To cluster analysis.* One of the problems in cluster analysis, for example in the method proposed in [7], is to find an efficient way of partitioning a set of n points in general position in $\mathbb{R}^d$ into two-disjoint subsets, separated by a hyperplane from one another [17]. This can be done in

$$\sum_{i=0}^{d} \binom{n-1}{i} \qquad (= 2^{n-1} \text{ for } n-1 \leqq d)$$

distinct ways [17], [21], and Harding [17] suggests that an algorithm for enumerating these partitions "without effectively considering *en route* the remainder of the 2^{n-1} associations into two sets" would be of value. Such an algorithm follows immediately from our results. For all one has to do, at least in the case Harding is interested in, when $P_1, \cdots, P_n$ are in general position, is to modify our sorting algorithm, Algorithm 3.1, by *listing* the points P_k on the positive side of each hyperplane $\langle P_{i_1}, \cdots, P_{i_{d-1}}, P_j \rangle$ instead of *counting* them (i.e. calculating Λ instead of λ), and then adding, in turn, each subset of $\{P_{i_1}, \cdots, P_{i_{d-1}}, P_j\}$ to the points in $\Lambda(i_1, \cdots, i_{d-1}, j)$; these will be all the semispaces, i.e. the subsets of $\mathscr{C}$ lying on one side of some hyperplane, and the repetitions will not affect the order of magnitude in n. This procedure will clearly generate *all* the partitions by hyperplanes, since each hyperplane can be moved continuously without crossing any point of $\mathscr{C}$ until it passes through d points of $\mathscr{C}$.

The following are some problems that this work suggests:

Problem 6.5. Prove (or disprove) Conjecture 6.1, and—if it is true—find the value of c.

Problem 6.6. Find a criterion for a generalized configuration in the plane to be realizable by points, i.e., to be equivalent to a configuration in $\mathbb{R}^2$. This problem, which will shed light on the question of the optimality of our algorithms, can be thought of as a special case of the coordinatizability problem for oriented matroids: What we have been calling an equivalence class of generalized configurations is also known [2], [4], [8] as an orientation class of acyclic oriented matroids of rank 3, and our λ-classification gives a classification of oriented matroids of *every* rank $d+1$, as well as of ordinary configurations in every dimension d. A matroid that corresponds to an ordinary configuration is said to be *coordinatizable over* $\mathbb{R}$, and it is a difficult and long-outstanding problem to characterize these among all oriented matroids. In particular, what proportion of generalized configurations are equivalent to ordinary configurations? (One can conjecture that the proportion will approach zero as the number of points increases.)

Problem 6.7. Find a fast algorithm for generating the lattice of faces of conv $(\mathscr{C})$ for a configuration $\mathscr{C}$ *not in general position* in $\mathbb{R}^d$, possibly by using $\lambda_{\mathscr{C}}$. Since the function $\lambda_{\mathscr{C}}$ carries all the information about $\mathscr{C}$ essential to this question, it should be possible to do this, optimally in time $O(n^{\lfloor d/2 \rfloor})$, since there are $O(n^{\lfloor d/2 \rfloor})$ faces, as we have seen. The result will be useful, among other things, in shortening the time needed to compute the set of canonical orderings of $\mathscr{C}$, hence for the comparison algorithm, in the case where the configuration $\mathscr{C}$ is not in general position.

Problem 6.8. Characterize the function $\lambda_{\mathscr{C}}$. What are its defining properties as a function on all ordered d-tuples chosen from an n-set? How does it behave with respect to subconfigurations, intersections, and so on? This is related, of course, to the axiomatic description of oriented matroids in [2] and [8], and perhaps even more to the (equivalent) axiomatic description of *chirotopes* in [5], but it is far from clear how properties of the matroid structure, which are essentially properties of the function Λ, will carry over to properties of the more compact function λ.

Acknowledgments. We would like to express our appreciation to A. Dress and G. Purdy for some helpful discussions, and to the referees for their suggestions on how best to present the algorithms.

REFERENCES

[1] J. K. AGGARWAL, R. O. DUDA, AND A. ROSENFELD, eds., *Computer Methods in Image Analysis*, IEEE Press, New York, 1977.

[2] R. G. BLAND AND M. LAS VERGNAS, *Orientability of matroids*, J. Combin. Theory Ser. B, 24 (1978), pp. 94–123.

[3] D. R. CHAND AND S. S. KAPUR, *An algorithm for convex polytopes*, J. Assoc. Comput. Mach., 17 (1970), pp. 78–86.

[4] R. CORDOVIL, *Sur les matroïdes orientés de rang trois et les arrangements de pseudodroites dans le plan projectif réel*, European J. Combin., to appear.

[5] A. S. DREIDING, A. DRESS AND H. R. HAEGI, *Chirotopes, a combinatorial theory of orientation*, preprint.

[6] A. S. DREIDING AND K. WIRTH, *The multiplex—a classification of finite ordered point sets in oriented d-dimensional spaces*, Match, 8 (1980), pp. 341–352.

[7] A. W. F. EDWARDS AND L. L. CAVALLI-SFORZA, *A method for cluster analysis*, Biometrics, 21 (1965), pp. 362–375.

[8] J. FOLKMAN AND J. LAWRENCE, *Oriented matroids*, J. Combin. Theory Ser. B, 25 (1978), pp. 199–236.

[9] J. E. GOODMAN AND R. POLLACK, *On the combinatorial classification of nondegenerate configurations in the plane*, J. Combin. Theory Ser. A, 29 (1980), pp. 220–235.

[10] ———, *Proof of Grünbaum's conjecture on the stretchability of certain arrangements of pseudolines*, J. Combin. Theory Ser. A, 29 (1980), pp. 385–390.

[11] ———, *Helly-type theorems for pseudoline arrangements in* $\mathbb{R}^2$, J. Combin. Theory Ser. A, 32 (1982), pp. 1–19.

[12] ———, *Semispaces of configurations, cell complexes of arrangements*, to appear.

[13] ———, *The λ-matrix: a new tool for pattern recognition*, in preparation.

[14] R. L. GRAHAM, *An efficient algorithm for determining the convex hull of a planar set*, Inform. Process. Lett., 1 (1972), pp. 132–133.

[15] B. GRÜNBAUM, *Convex Polytopes*, Interscience-Wiley, London, 1967.

[16] ———, *Arrangements and Spreads*, CBMS Regional Conference Series in Applied Mathematics 10, American Mathematical Society, Providence, RI, 1972.

[17] E. F. HARDING, *The number of partitions of a set of n points in k dimensions induced by hyperplanes*, Proc. Edinburgh Math. Soc., 15 (1967), pp. 285–289.

[18] D. E. KNUTH, *The Art of Computer Programming, Vol. 3, Sorting and Searching*, Addison-Wesley, Reading, MA, 1973.

[19] F. P. PREPARATA AND S. J. HONG, *Convex hulls of finite sets of points in two and three dimensions*, Comm. Assoc. Comput. Mach., 20 (1977), pp. 87–93.

[20] M. I. SHAMOS, *Computational geometry*, Ph.D. thesis, Yale Univ., New Haven, CT, 1978.

[21] T. ZASLAVSKY, *Facing Up to Arrangements: Face-Count Formulas for Partitions of Space by Hyperplanes*, Memoir 154, American Mathematical Society, Providence, RI, 1975.