DIMACS Technical Report 94-11
April 1994

A pseudo-algorithmic separation of lines from pseudo-lines

by

William Steiger[1,2]
Dept. of Computer Science
Rutgers University
New Brunswick, New Jersey 08903

Ileana Streinu[3]
Dept. of Computer Science
Rutgers University
New Brunswick, New Jersey 08903

[1]Permanent Member
[2]Research Supported in Part by NSF grant CCR-9111491
[3]Affiliated Graduate Student Member

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

The x-sorting problem for lines is: sort the x-coordinates of the intersection points of n planar lines. A similar problem can be defined for pseudo-lines. We show a lower bound of $\Omega(n^2 \log n)$ for x-sorting pseudo-lines and the existence of a quadratic decision tree of depth $\mathcal{O}(n^2)$ for x-sorting lines. Let X and Y be two n element sets and let $X + Y = \{x + y \mid x \in X, y \in Y\}$. Sorting $X + Y$ is a particular case of x-sorting lines. Fredman [4] has shown the existence of an $\mathcal{O}(n^2)$-depth decision tree for this case and Lambert [9] has given a method to actually find the $\mathcal{O}(n^2)$ comparisons. Here we give another very simple divide and conquer algorithm for sorting $X + Y$ with $\mathcal{O}(n^2)$ comparisons which only needs $\mathcal{O}(n^2 \log n)$ time.

1 Introduction

Let $L = \{\lambda_1, \cdots, \lambda_n\}$ be a collection of n continuous functions on R with the property that for each pair $i \neq j$ there is an $x = x_{ij}$ such that $\lambda_i(x_{ij}) = \lambda_j(x_{ij})$ and $(\lambda_i(t) - \lambda_j(t))(\lambda_i(u) - \lambda_j(u)) < 0$ whenever $u < x_{ij}$ and $t > x_{ij}$. L is a collection of *pseudo-lines*. In the special case where the λ_i are linear functions we have a set of distinct *lines*, no two parallel. Let

$$S = \{x_{ij} : i < j\}$$

be the set of the x-coordinates of the vertices $\lambda_i \cap \lambda_j$ of the collection.

L induces a decomposition of the plane into regions bounded by edges and vertices. This decomposition, $A(L)$, is called the arrangement of the pseudo-lines. It is known that pseudo-line arrangements have a richer combinatorial structure than arrangements of lines: the number of combinatorially different pseudo-line arrangements is much larger than the number of line arrangements, as shown by Goodman and Pollack [6].

It would be of great interest to find a geometric property (or another combinatorial property besides cardinality) whose combinatorial expression could be used to separate line and pseudo-line arrangements. The complexity of the k-th level (number of k-sets) is believed to be such a property, though with the present knowledge no such separation can be claimed (see [11]). The same may be said about the complexity of an x-monotone path (see [10]).

In this paper we give an algorithmic-type of separation for these two classes via the complexity of sorting the elements of S, the x-coordinates of the vertices of the arrangement. We call this the x-sorting problem for L. Using ideas from Goodman and Pollack[6] (or from a straightforward reduction argument) it is necessary to make $\Omega(n^2 \log n)$ comparisons to sort S for pseudo-lines; by applying Fredman's result [4] on sorting under partial information, there exists a quadratic decision tree of depth $\mathcal{O}(n^2)$ to sort the vertices of a line arrangement. This separation is called *pseudo-algorithmic* because it holds in the decision tree model, which is a non-uniform model of computation. A truly algorithmic separation is not known: no polynomial time way of actually *constructing* the comparisons in an algebraic decision tree of depth $\mathcal{O}(n^2)$ is known.

Although the idea of applying Fredman's result to x-sorting may not be new, we have not been able to find it in the literature. Neither does the separation observation seem to have been made. We think that this is the first example of a problem for which getting an optimal algorithm would require the use of *some* property that holds for lines but not for pseudo-lines. As many algorithms in Computational Geometry use only combinatorial properties of line arrangements they work for pseudo-lines as well. We think this is an interesting situation, where a *geometric* distinction is needed.

x-sorting for lines has an interesting special case, the $X + Y$ problem: given sets X and Y of n numbers each, sort the n^2 sums $\{x_i + y_j | x_i \in X, y_j \in Y\}$. In [4], Fredman showed that there exists a decision tree of depth $\mathcal{O}(n^2)$ for sorting $X + Y$. Later, Lambert [9] gave an algorithm to actually get such a quadratic depth tree. Lambert did not give a RAM implementation for his algorithm, but it seems to incur a large bookeeping cost. In section 3 we will present a much simpler algorithm which will construct the $\mathcal{O}(n^2)$ comparisons to sort

$X + Y$ and use only $\mathcal{O}(n^2 \log n)$ total time. It is still an open problem to find an algorithm for sorting $X + Y$ in $o(n^2 \log n)$ *time* and we think that our approach may eventually lead to a solution to this long standing question.

2 Main Result

Given a collection $L = \{\lambda_1, \ldots, \lambda_n\}$ of n pseudo-lines with vertices $\lambda_i \cap \lambda_j$ we make the *general position* assumption that no point meets three pseudo-lines. This implies that in a suitable coordinate system the set $S = \{x_{ij} : i < j\}$ has $\binom{n}{2}$ distinct elements. The x-sorting problem for L is to order the elements of S.

Theorem 1 *There exists a quadratic algebraic decision tree of depth $\mathcal{O}(n^2)$ that does x-sorting for lines. The depth of any decision tree that does x-sorting for pseudo-lines is $\Omega(n^2 \log n)$.*

Proof: The proof of the upper bound for lines is based on the following result

Theorem 2 *(Fredman[4]) There exists a decision tree of depth at most $\log |P| + 2N$ which solves the problem of sorting under partial information for a set X of N elements, with partial information from the set P.*

Here $X = \{x_1, \cdots, x_N\}$ is an N element set and P is a subset of the $N!$ possible linear orderings on X. The problem of *sorting under partial information* is to identify an unknown ordering $\omega \in P$ by performing comparisons between the elements of X. A decision tree is said to solve the problem of sorting under partial information for X if it has a leaf for each $\omega \in P$. We will be interested in the depth of the optimal decision tree which solves the sorting problem for X.

We apply this theorem when $X = S$, the set of x-coordinates of the $N = \binom{n}{2}$ vertices of the given n lines and P, the set of all possible sorted orders of S. The x-coordinate of the intersection of two lines $y = a_i x + b_i$ and $y = a_j x + b_j$ is

$$x_{ij} = \frac{b_i - b_j}{a_j - a_i}.$$

Assuming that we have already sorted the slopes $a_1, \ldots, a_n$ (in $\mathcal{O}(n \log n)$ time), we know the sign of the denominators. Thus the comparison between x_{ij} and x_{kl} can be transformed into a comparison between $(b_k - b_l)(a_i - a_j)$ and $(b_i - b_j)(a_k - a_l)$, each product appropriately adjusted for sign. The latter comparison is achieved in a quadratic algebraic decision tree. Finally, $|P|$ can be estimated using a result of Goodman and Pollack [6]. Once the lines are in the order of decreasing slope, their theorem implies that $|P| = \mathcal{O}(n^{8n})$. We get the upper bound for x-sorting lines by plugging this into Fredman's result.

The lower bound for x-sorting pseudo-lines is information-theoretic. The number of x-sorted orders of the vertices of an arrangement of n pseudo-lines ν is given by the following

precise formula, independently obtained by Edelsman and Greene[2] and Stanley[12] (cf. Goodman and Pollack [6]):

$$\nu = \frac{(n-2)!\binom{n}{2}!}{1^{n-1}3^{n-2}\cdots(2n-3)^1}$$

The logarithm of this is asymptotically $cn^2 \log n$. It follows that any decision tree for x-sorting pseudo-lines has depth $\Omega(n^2 \log n)$. ∎

Note that the lower bound for pseudo-lines is tight since the $\mathcal{O}(n^2)$ intersection points of n pseudo-lines can be sorted in $\mathcal{O}(n^2 \log n)$ time by any optimal sorting algorithm. For x-sorting lines the situation is different. Information theoretically (i.e. in the decision tree model) the upper bound doesn't match the lower bound, which is just $\Omega(n \log n)$. It is an open question whether one can do x-sorting for n lines with only $\mathcal{O}(n \log n)$ comparisons or whether a better lower bound is possible. Neither do the bounds match in the general (RAM) model of computation, where a lower bound of $\Omega(n^2)$ can be obtained just from the complexity of writting down the sorted list of vertices. It's an open problem to do x-sorting for lines in $o(n^2 \log n)$ total time.

Remark: The lower bound to x-sorting for pseudo-lines can also be obtained easily without using the precise formula given above. We can reduce the problem of sorting the entries of an n by n ordered matrix A (all rows and all columns are non-decreasing) to x-sorting the vertices of $2n$ pseudo-lines . Then we use the lower bound of Harper et al. [8]: sorting an ordered matrix of size n requires $\Omega(n^2 \log n)$ time. For the reduction, given A with nondecreasing rows and columns, we want to construct n pseudo-lines for which $x_{ij} = A(i,j)$. Just define points $P_{ij} = (x_{ij}, j-i)$, for $i,j = 1,\ldots,n$ and let $I = (x_{11}, x_{nn})$. For each i, define (row) pseudo-line r_i as the piecewise linear function joining the P_{ij} , $j = 1,\ldots,n$ and for each j define (column) pseudo-line c_j as the piecewise linear function joining P_{ij} , $i = 1,\ldots,n$. The r_i are non-decreasing and the c_j are non-increasing. Also, $r_i(P_{ij}) = c_j(P_{ij})$ but there are no other incidences. Finally the r_i and c_j can be extended to plus and minus ∞ so the r_i are increasing and each pair has a proper intersection outside I and so the c_j are decreasing and each pair has a proper intersection outside I. x-sorting for these pseudo-lines will order the entries of A and the reduction can be done in $\mathcal{O}(n^2)$ steps.

3 Sorting $X + Y$

Given $X = \{x_1, \cdots, x_n\}$ and $Y = \{y_1, \cdots, y_n\}$, define the lines $y = x_i, i = 1, \cdots, n$ and $y = x - y_j, j = 1, \cdots, n$. The x-coordinates of the intersection points are the elements of the set $X + Y$ (and because of parallelism, two degenerate points at infinity). This shows that sorting the cartesian sums $X + Y$ is a particular case of the x-sorting problem for lines. Without loss of generality we may take X and Y positive.

As in Lambert [9], we reduce the $X + Y$ problem to the problem of sorting *interval sums*.

For a set of m positive numbers $a_1, \cdots, a_m$, the $\binom{m}{2}$ interval sums are defined to be

$$\sigma_{ij} = \sum_{k=i}^{j} a_k, \ i < j. \tag{1}$$

For the reduction, given X and Y, we sort them so $x_1 \leq \cdots \leq x_n$ and $y_1 \leq \cdots \leq y_n$. Now define $a_n = x_1$ and $a_{n-i} = x_{i+1} - x_i$, $i < n$; also set $a_{n+1} = y_1$ and $a_{n+j+1} = y_{j+1} - y_j, j < n$. It follows that

$$x_i + y_j = \sum_{k=n-i+1}^{n+j} a_k = \sigma_{n-i+1,n+j}.$$

The reduction uses quadratic time to obtain the interval sums. Once the σ_{ij} are sorted, so are the sums $X + Y$.

At this point remember what we mean by *sorted* in the two computational models that we are using: the decision tree and the RAM model. In the decision tree model, only comparisons between σ_{ij}'s are counted and they cost one unit. Each answer splits the set of possible total orders compatible with the answers received so far into two subsets. We say that a sequence of comparisons has sorted the interval sums if the set of comparisons has a unique permutation compatible with the results of those comparisons. In the RAM model, by *sorted* we mean that we have $\rho(\sigma_{ij})$, the rank of each of the $N = \binom{2n}{2}$ interval sums and also, for each $k = 1, \ldots, N$, that we know the σ_{ij} with $\rho(\sigma_{ij}) = k$ (or equivalently, that the interval sums are arranged in a linear array $L_1 \leq \cdots \leq L_N$).

Theorem 3 *The interval sums $S = \{\sigma_{ij} = a_i + \cdots + a_j, i < j\}$ over a set of n positive numbers $a_1, \ldots, a_n$ may be computed in $\mathcal{O}(n^2)$ comparisons (between σ_{ij}'s) in time $\mathcal{O}(n^2 \log n)$.*

Proof: We describe a RAM algorithm with the stated complexity. Let T be the partial order over S induced by the relations

$$\sigma_{i,j-1} \leq \sigma_{ij} \quad \text{and} \quad \sigma_{ij} \geq \sigma_{i+1,j}.$$

We can represent T as the *lattice poset* in Figure 1, with triangular shape, σ_{1n} on top, and σ_{ij} having left child $\sigma_{i,j-1}$ and right child $\sigma_{i+1,j}$.

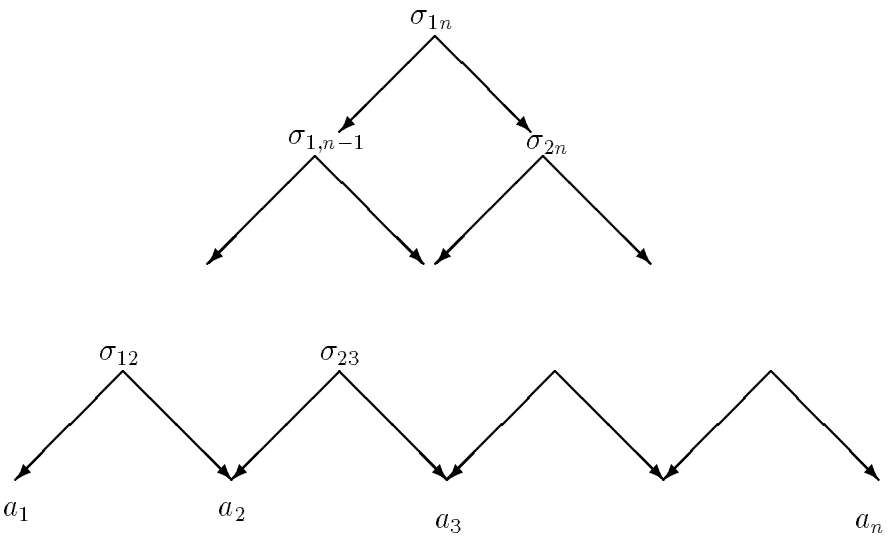

Fig.1. *The lattice poset associated with the interval sums over* $a_1, \cdots, a_n$

The algorithm uses a divide and conquer approach. Assuming n is even we split T into two smaller triangles T_1 and T_2 and a diamond D (see Figure 2):

$$T_1 = \{\sigma_{ij},\ i < j \leq \frac{n}{2}\};\ \ T_2 = \{\sigma_{ij},\ \frac{n}{2} < i < j\};$$

$$D = \{\sigma_{ij},\ i \leq \frac{n}{2},\ j > \frac{n}{2}\}.$$

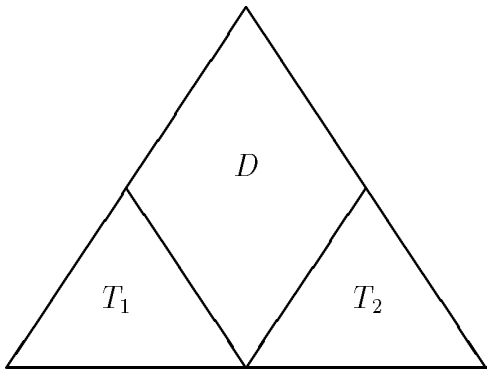

Fig.2. *Sorting interval sums in* T *recursively.*

It is used as follows.

Algorithm SORT(T)

1. Split T into T_1, T_2, and D.
2. Run SORT(T_1) and SORT(T_2).
3. Sort $T_1 \cup T_2$ by merging.
4. Sort D
5. Merge D and $T_1 \cup T_2$.

Step 2 applies the present algorithm recursively to the elements in T_1 and T_2. When it is completed, the interval sums from T_1 are in a sorted array L_1 and those from T_2, in a sorted array L_2. Step 3 merges these arrays into the sorted array L.

To sort the n^2 elements in D we follow the control of any optimal $\mathcal{O}(n^2 \log n)$ sorting algorithm. A typical step will ask for comparisons between σ_{rs} and σ_{uv}, elements of D. We may assume $r < u$. If also $s \geq v$ then by (1) and additivity, $\sigma_{rs} \geq \sigma_{uv}$, and no work is required for this comparison. Otherwise note that

$$\sigma_{rs} - \sigma_{uv} = \sum_{i=r}^{u-1} a_i - \sum_{i=s+1}^{v} a_i = \sigma_{r,u-1} - \sigma_{s+1,v}. \tag{2}$$

Also $\sigma_{r,u-1} \in T_1$ and $\sigma_{s+1,v} \in T_2$. Since $T_1 \cup T_2$ is already sorted, the sign of the difference of these two interval sums is obtained at no cost in the comparison model, and at unit cost in the RAM model, namely the cost of comparing $\rho(\sigma_{ru})$ with $\rho(\sigma_{sv})$; the whole cost of step 4 is $\mathcal{O}(n^2 \log n)$ in the RAM model.

Let $C(n)$ be the comparison complexity of SORT(T) when T represents interval sums of $a_1 \leq \cdots \leq a_n$, and $R(n)$, the RAM complexity. Therefore

$$R(n) \leq \frac{n^2}{2} + 2R(\frac{n}{2}) + \frac{n^2}{4} + \alpha n^2 \log n + \frac{n^2}{2}.$$

The five terms on the right are the costs of the above steps. The relation is satisfied by $R(n) = \mathcal{O}(n^2 \log n)$.

From the RAM algorithm SORT it is straightforward to construct a decision tree for a fixed n. Here only steps 2,3 and 5 need to be considered. Step 4 may be ignored, since by (2) there is a unique permutation of the $\sigma_{ij} \in D$ that is compatible with the merge in step 3. This means that:

$$C(n) \leq 2C(\frac{n}{2}) + \frac{n^2}{4} + \frac{n^2}{2}.$$

The first term is the cost of step 2, the next of step 3, and the last of step 5. We get $C(n) = \mathcal{O}(n^2)$. ∎

4 Final Remarks

For x-sorting vertices in line arrangements, it is not even known how to construct a quadratic depth decision tree whose existence is guaranteed by Theorem 1. We do not know how to do this even for the particular case when the n lines are the duals of n points in convex position. This arrangement has a simple lattice structure, so it is as close to the structure of the $X + Y$ problem that we can get with general lines. However we do know that to carry out such a construction it will be necessary to utilize geometric properties - and their combinatorial expression - that are not satisfied by general pseudo-lines .

Acknowledgement: We thank Hari Hampapuram for valuable conversations.

References

[1] R.Cole, J. Salowe, W. Steiger and E. Szemerédi, An Optimal-time algorithm for Slope Selection, SIAM Journal on Computing 18, 792-810, 1989

[2] P. Edelman and C. Greene, *Combinatorial correspondences for Young tableaux, balanced tableaux, and maximal chains in the Bruhat order of* S_n, in *Combinatorics and Algebra, Contemporary Math.*, vol. 34, American Mathematical Society, 1984.

[3] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag (1987).

[4] M. Fredman, *How good is the Information Theory bound in Sorting?*, Theoretical Computer Science 1, pp. 355-361, 1976.

[5] J.E. Goodman and R. Pollack, *Multidimensional sorting*, SIAM J. Comput. vol. 12, No.3, August 1983.

[6] J.E. Goodman and R. Pollack, *Upper Bounds for Configurations and Polytopes in* R^d, Discrete Comput. Geom. $1 : 219 - 227$,1986.

[7] J.E. Goodman and R. Pollack, *Allowable Sequences and Order Types in Discrete and Computational Geometry*, DIMACS Tech. Report 92-1; in: J.Pach (ed.), *New trends in Discrete and Computational Geometry*,1991.

[8] L.H. Harper, T.H. Payne, J.E. Savage and E. Strauss, *Sorting* $X + Y$, Comm. ACM, vol.18, no.6, pp.$347 - 349$, 1975.

[9] J.L. Lambert, *Sorting* $X + Y$ *in* $O(n^2)$ *comparisons*, STACS 1991, pp.195-206.

[10] J. Matoušek, *Lower Bounds on the Length of Monotone Paths in Arrangements*, Discrete and Comp.Geom. 6, 129-134, 1991.

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

[12] R.Stanley, *On the Number of Reduced Decompositions of Elements of Coxeter Groups*, Europ. J. Combinatorics 5, pp. 359 – 372, 1984.