On the Relative Complexities of Some Geometric Problems*

(Extended Abstract)

Jeff Erickson†

1 Introduction

We consider a number of problems whose best known algorithms run in roughly $O(n^{4/3})$ time. While it is generally believed that these algorithms are optimal, at least up to polylogarithmic or n^{ε} factors, the best known lower bound in any general model of computation is only $\Omega(n \log n)$. We characterize these problems by their *relative* complexities. That is, for certain pairs of problems, we show that the complexity of one problem is asymptotically bounded by the complexity of the other. Thus, a $o(n^{4/3})$-time algorithm for the "harder" problem is impossible without a similar algorithm for the "easier" one. Conversely, any better lower bounds for the easier problem would immediately apply to the harder problem as well.

This paper is similar in spirit to the earlier work of Gajentaan and Overmars [17]. They introduce the class of 3SUM-hard problems, all of which are at least as hard as the following simple base problem: Given n numbers, do any three sum to zero? All of these problems seem to require $\Omega(n^2)$ time to solve; thus, some earlier papers describe them with the more suggestive but misleading term "n^2-hard". (See [6].) The present classification of "$n^{4/3}$-hard" problems is not so clean. Many of our reductions introduce extra logarithmic factors. More importantly, we do not know if there is a single base problem to which all these problems can be reduced.

2 Definitions and Background

2.1 Known Upper and Lower Bounds

For each of the problems we consider, the best lower bound known in any general model of computation is only $\Omega(n \log n)$. This lower bound follows from results of Steele and Yao in the algebraic decision tree model [25] and Ben-Or in the algebraic computation tree model [4].

Better lower bounds are known for a few of these problems in less powerful models, but these models are inappropriate in more general settings. Chazelle [8, 7] has proven a number of lower bounds for online and offline range counting problems in the Fredman/Yao semigroup arithmetic model [16]. While this model works quite well for studying this sort of counting problem, it is not at all applicable to decision or optimization problems. Similarly, Erickson [15] has proven $\Omega(n^{4/3})$ lower bounds for a number of problems, including Hopcroft's problem (Problem A) and unit distance detection (Problem F), in what he calls the partitioning algorithm model. This model is specifically tailored towards a specific class of incidence-detection problems. Some problems, such as halfspace range checking (Problem E), can be solved in linear time in this model. Others, such as line towering (Problem J), apparently cannot be solved at all.

2.2 Relative Complexity

Let $T_1(n)$ and $T_2(n)$ denote the complexities of two problems Π_1 and Π_2, expressed as functions of the input size n. If $T_1(n) = \Omega(T_2(n))$, or equivalently, if $T_2(n) = O(T_1(n))$, we say that **Π_1 is harder than Π_2**. If $T_2(n) = O(T_1(n) \log^c n)$ for some constant c, we say that **Π_1 is almost harder than Π_2**. Typically, this means that Π_2 is solved by a binary search or parametric search [13, 22], using an algorithm for Π_1 as an oracle, although more complicated reductions are also possible. Finally, we say that **Π_1 is probably harder than Π_2** if the following conditions are met.

1. Π_1 is almost harder than Π_2.
2. If $T_1(n) = \Omega(n^{1+\varepsilon})$ for some $\varepsilon > 0$, then Π_1 is harder than Π_2.

For example, suppose Π_2 is solved using a recursive divide-and-conquer strategy, similar to quicksort or mergesort: the input is divided into two halves in linear time, an algorithm for Π_1 is applied to the partitioned input, and problem Π_2 is recursively solved for each of the two partitions. Then we have the following upper bound for $T_2(n)$ in terms of $T_1(n)$.

$$T_2(n) = O(T_1(n)) + 2T_2(n/2) = \sum_{i=0}^{\lceil \lg n \rceil} 2^i O(T_1(n/2^i))$$

*Portions of this research were done while the author was visiting Universität des Saarlandes, Freie Universität Berlin, and Universiteit Utrecht.

†Computer Science Division, University of California, Berkeley, CA 94720. Email: jeffe@cs.berkeley.edu

This upper bound can always be simplified to $T_2(n) = O(T_1(n)\log n)$, but if $T_1(n) = \Omega(n^{1+\varepsilon})$ for some constant $\varepsilon > 0$, the bound improves to $T_2(n) = O(T_1(n))$.

2.3 Infinitesimal Reductions

We derive our relative complexity results through the standard technique of reductions. Suppose we want to show that problem Π_1 is harder than problem Π_2. In the simplest reduction argument, given an instance of Π_2, we transform it into an instance of Π_1 of roughly the same size, call an algorithm for Π_1 as a subroutine, and transform the output of this subroutine to the result of Π_2.

We develop our relative complexity results in the algebraic decision tree model of computation [25]. Most of our results hold in more general models of computation such as algebraic computation trees [4] or the real RAM [24], but a few results rely on specific properties of the algebraic decision tree model.

Specifically, some of our reductions introduce formal infinitesimals into the input before passing it to the Π_1 subroutine. See, for example, the proof of Theorem 4. In order to justify this practice, we need two technical lemmas. We will give only a cursory sketch of these lemmas in this extended abstract; the formal statements and proofs are included in the full version of the paper.

The first problem is that we are given an algorithm for Π_1 that works on any real input, but we need an algorithm that works even if the input contains infinitesimals. In the full paper, we show that any algorithm that solves a problem over the reals also solves the same problem over any real closed field. For a similar argument, see [14].

The second problem is that an algebraic decision tree cannot *really* introduce infinitesimals into the input, since the model only allows query polynomials with real coefficients. Thus, we must show that the infinitesimals can be *simulated* as follows. Represent each "number" $a+b\varepsilon+c\varepsilon^2+\cdots$ as a polynomial in ε, and modify the Π_1 subroutine to perform polynomial arithmetic in place of real arithmetic. Since the polynomials introduced by the transformation have bounded degree, and the query polynomials used by algebraic decision trees have bounded degree, this modification increases the running time of the algorithm by only a constant factor. We cannot similarly modify algebraic computation trees or real RAMs, since algorithms in these models can compute polynomials of unbounded degree.

3 Hopcroft's Problem and its Friends

We begin with one of the oldest problems in computational geometry, first posed by John Hopcroft in the early 1980's.

Problem A. Hopcroft's Problem: Given a set of points and lines in the plane, does any point lie on any line?
Best known upper bound: $O(n^{4/3}2^{O(\log^* n)})$ [21]

Hopcroft's problem is a special case of a large number of other more general problems, including the following.

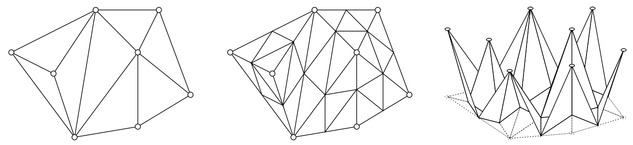
- Detecting, counting, or enumerating incidences between a set of "point-like" geometric objects (points, line segments, circles, triangles, etc.) and a set of "line-like" geometric objects (lines, line segments, rays, circles, etc.)
- Finding the closest point-line pair
- Locating a set of points in a line arrangement
- Counting intersections among line segments
- Triangular range checking (Does any triangle contain a point?)
- Halfplane range counting (How many points are in each halfplane?)
- Detecting intersections, or finding the closest pair, among lines in $\mathbb{R}^3$

Rather than attempting to give an exhaustive list of easy reductions, we will describe only a few specific problems, for which the reductions may be less obvious.

Problem B. Ray Shooting over a Polyhedral Terrain: Given a polyhedral terrain and a set of rays in $\mathbb{R}^3$, does any ray hit the terrain?
Best known upper bound: $O(n^{4/3+\varepsilon})$ [10]

Theorem 1. *Ray shooting over a polyhedral terrain is harder than Hopcroft's problem.*

Proof: Given a set of points and lines, we construct a polyhedral terrain and a set of rays as follows. Construct an arbitrary triangulation of the points. Divide each triangle into four sub-triangles by bisecting its edges. See Figure 1(b). This ensures that no triangle shares two of the original points. To create the terrain, lift the triangulation into $\mathbb{R}^3$, placing each of the original points on the plane $z = 1$, and all the other vertices on the plane $z = 0$. See Figure 1(c). To create the rays, chop each line at some very large x- or y-coordinate, and lift the resulting ray to the plane $z = 1$. A ray hits the terrain if and only if the corresponding line contains one of the original points. The entire reduction can be carried out in time $O(n \log n)$. $\square$



(a) (b) (c)

Figure 1. Transforming Hopcroft's problem into ray shooting over a polyhedral terrain (Theorem 1). (a) Initial triangulation of the points. (b) Refined triangulation. (c) Final terrain.

Problem C. Segment Depth Order Verification: Given a sequence of non-intersecting line segments, is any segment below a segment following it in the sequence?
Best known upper bound: $O(n^{4/3+\varepsilon})$ [5]

Theorem 2. *Segment depth order verification is harder than Hopcroft's problem.*

Proof: Suppose we are given a set $\{p_1, \ldots, p_n\}$ of points and a set $\{l_1, \ldots, l_n\}$ of lines. Lift each point p_i onto the horizontal plane $z = -i$ and each line l_j onto the plane $z = j$. Present the sequence $(p_1, p_2, \ldots, p_n, l_1, l_2, \ldots, l_n)$ to a depth order verification algorithm, where each point is considered a segment of zero length and each line a segment of infinite length. The depth order algorithm reports that some pair of "segments" is out of order if and only if there is a point-line incidence in the original input. □

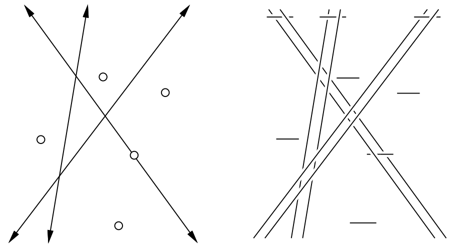
Problem D. Segment Cyclic Overlap: Given a set of non-intersecting line segments in $\mathbb{R}^3$, does any subset overlap cyclically?
Best known upper bound: $O(n^{4/3+\varepsilon})$ [5]

Theorem 3 (De Berg *et al.* [5]). *Segment depth order verification is harder than segment cyclic overlap.*

Proof: The segment cyclic overlap problem can be decided by applying any $O(n \log n)$ sorting algorithm to the segments, and checking to see if the resulting sequence is actually sorted. □

Theorem 4. *Segment cyclic overlap is harder than Hopcroft's problem, in the algebraic decision tree model of computation.*

Proof: Suppose we are given a set $P = p_1, p_2, \ldots, p_n$ of points and a set $L = l_1, l_2, \ldots, l_n$ of lines. Without loss of generality, none of the lines is horizontal. We produce a set of non-intersecting segments as follows. Replace each line l_i with two parallel lines l_i^+ and l_i^-, at horizontal distance ε to the right and left of l_i, respectively, where ε is a formal infinitesimal. Lift each line l_i^+ to the plane $z = 3i + 1$ and each line l_i^- to the plane $z = 3i - 1$.



(a) (b)

Figure 2. Transforming Hopcroft's problem into segment cyclic overlap (Theorem 4). (a) Initial set of points and lines. (b) Transformed set of segments, seen from above. The incidence is transformed into an overlap cycle.

Next, for each line l_i, add a segment s_i parallel to the x-axis and in the plane $z = 3i$, such that the projection of s_i intersects the projections of both l_i^+ and l_i^-, but is far away from every other point and line. It suffices to put s_i at an y-coordinate far above all the original points and line intersections.

Finally, replace each point $p_j = (x_j, y_j)$ with a line segment $\hat{p}_j$ with endpoints $(x_j - 2\varepsilon, y_j, -3n)$ and $(x_j + 2\varepsilon, y_j, 6n)$. These line segments are almost parallel to the z-axis. Our final collection of $4n$ line segments consists of l_i^+, l_i^-, s_i, and $\hat{p}_j$ for all i, j. The entire transformation can be performed in time $O(n \log n)$.

If there are no incidences between P and L, then none of the segments $\hat{p}_j$ are above or below any other segment. Since the other segments are all parallel to the xy-plane, there are no cyclic overlaps. On the other hand, if there is an incidence between l_i and p_j, then the four segments $\hat{p}_j, l_i^+, l_i^-, s_i$ overlap cyclically. See Figure 2. □

Problem E. Halfspace Range Checking in $\mathbb{R}^5$: Given a set of points and hyperplanes in $\mathbb{R}^5$, is every point above every hyperplane?
Best known upper bound: $O(n^{4/3} \log^{O(1)} n)$ [19]

Theorem 5. *Halfspace range checking in $\mathbb{R}^5$ is harder than Hopcroft's problem.*

Proof: Every point and hyperplane in $\mathbb{R}^d$ can be represented in homogeneous coordinates by a vector in $\mathbb{R}^{d+1}$, so that the relative orientation of any point p and any hyperplane h is determined by the sign of the inner product $\langle p, h\rangle$. Specifically, if $\langle p, h\rangle > 0$ then p is above h; if $\langle p, h\rangle = 0$, then p is contained in h; and if $\langle p, h\rangle < 0$, then p is below h.

Using this observation, we can reformulate the two problems as follows.

- Hopcroft's problem: Given a set of red and blue vectors in $\mathbb{R}^3$, is there a red vector p and a blue vector h such that $\langle p, h\rangle = 0$?

- Halfspace range checking in $\mathbb{R}^5$: Given a set of red and blue vectors in $\mathbb{R}^6$, is there a red vector p and a blue vector h such that $\langle p, h\rangle \leq 0$?

Now consider the function $\sigma : \mathbb{R}^3 \to \mathbb{R}^6$ defined as

$$\sigma(x, y, z) = (x^2, y^2, z^2, \sqrt{2}xy, \sqrt{2}yz, \sqrt{2}zx).$$

This function squares inner products: $\langle\sigma(p), \sigma(h)\rangle = \langle p, h\rangle^2$. Thus, we can transform any set of points and lines into a set of of five-dimensional points and hyperplanes in linear time, by applying the function σ to the entire set, so that no point is below any hyperplane and all incidences are preserved. $\square$

4 Planar Distance Problems

Problem F. Unit Distance Detection: Given a set of points in the plane, is any pair of points at unit distance?
Best known upper bound: $O(n^{4/3} \log^{2+\varepsilon} n)$ [18].

Erickson's lower bound for the unit distance problem follows immediately from his lower bound proof for Hopcroft's problem [15]. Nevertheless, we are unable to show that detecting unit distances is harder, or easier, than detecting point-line incidences, nor are we able to show that both are harder than some third simpler problem. Both problems are special cases of several other problems, such as point-circle incidence detection in the plane, point-plane incidence detection in $\mathbb{R}^3$, and unit-distance detection in $\mathbb{R}^3$.

Problem G. Distance Selection: Given a set of points in the plane and an integer k, what is the kth smallest interpoint distance?
Best known upper bound: $O(n^{4/3} \log^{3+\varepsilon} n)$ [18]

Theorem 6. *Distance selection is almost harder than unit distance detection.*

Proof: We can detect unit distances with a binary search over the $\binom{n}{2}$ possible values for k, using a distance selection algorithm at each step in the search. $\square$

Problem H. Distance Ranking: Given a set of points in the plane how many pairs of points are closer than unit distance apart?
Best known upper bound: $O(n^{4/3} \log^{3+\varepsilon} n)$ [18]

Theorem 7. *Distance selection is almost harder than distance ranking. Distance ranking is almost harder than distance selection.*

Proof: The binary search algorithm described in the proof of Theorem 6 can also be used to solve the distance ranking problem. To select the kth smallest distance, we can perform a parametric search over the space of interpoint distances, using a distance ranking algorithm as an oracle [2]. $\square$

Theorem 8. *Distance ranking is harder than unit distance detection in the algebraic decision tree model.*

Proof: To detect the presence or absence of unit distances in a set of points, we call a distance ranking algorithm twice, once on the original set of points, and once on the set of points scaled by a factor of $1 + \varepsilon$, where ε is an infinitesimal. The unit distance rank is the same in both sets if and only if the original set contains no unit distances. $\square$

5 Lines and Segments in Space

Problem I. Polyhedral Terrain Intersection: Given two polyhedral terrains in $\mathbb{R}^3$, do they intersect?
Best known upper bound: $O(n^{4/3} \log^{O(1)} n)$ [10, 9, 19]

Problem J. Line Towering: Given a set of red and blue lines in $\mathbb{R}^3$, are all the red lines above all the blue lines?
Best known upper bound: $O(n^{4/3} \log^{O(1)} n)$ [9, 19]

Theorem 9. *Line towering is almost harder than polyhedral terrain intersection.*

Proof: Chazelle *et al.* [10] show that the smallest vertical distance between two polyhedral terrains of total complexity n can be reduced to several instances of line towering, with total complexity $O(n \log^2 n)$. Their algorithm can also be used to decide if two terrains intersect. Since the complexity of line towering is $o(n^2)$, this reduction introduces a multiplicative factor of less than $\log^4 n$ into the running time. $\square$

Problem K. Line Cyclic Overlap: Given a set of non-intersecting lines in $\mathbb{R}^3$, do any three lines overlap cyclically?
Best known upper bound: $O(n^{4/3} \log^{O(1)} n)$ [11, 9, 19]

This problem is clearly a special case of segment cyclic overlap (Problem D).

Theorem 10 (Chazelle *et al.* [11]). *Line towering is almost harder than line cyclic overlap.*

Proof: The line cyclic overlap problem can be solved by applying any sorting algorithm to the lines, and checking whether the resulting sequence is a valid depth order. (Compare Theorem 3.) The sequence can be verified as follows. Split the sequence of lines into two equal halves, verify each of the two halves recursively, and use a line towering algorithm to check that every line in the first half is above every line in the second half. □

Problem L. Farthest Line Pair: Given a set of lines in $\mathbb{R}^3$, find the pair of lines separated by the largest vertical distance.
Best known upper bound: $O(n^{4/3} \log^{O(1)} n)$ [23, 9, 19]

Theorem 11 (Pellegrini [23]). *Line towering is almost harder than farthest line pair.*

Proof: The farthest line pair problem can be solved as follows. Split the set of lines into two classes, and recursively find the farthest line pair within each class. To find the farthest pair between the two subsets, perform a parametric search, using a line towering algorithm as an oracle. See [23] for further details. □

Theorem 12 (Chazelle *et al.* [9]). *Halfspace range checking in $\mathbb{R}^5$ is probably harder than line towering.*

Proof: Consider the special case of *consistently oriented* sets of lines, in which the projections of the blue lines onto the xy-plane all have higher slope than the projections of the red lines. In this case, using Plücker coordinates [26], we can express each red line as a point in $\mathbb{R}^5$ and each blue line as a hyperplane in $\mathbb{R}^5$, so that relative orientation is preserved.

The general problem can be solved using the following divide-and-conquer approach. The median slope among the xy-projections of all the lines naturally partitions the red lines into two subsets R_1 and R_2, and the blue lines into B_1 and B_2, so that the projected slopes in $R_1 \cup B_1$ are all larger than the projected slopes of $R_2 \cup B_2$. The subset pairs (R_1, B_2) and (R_2, B_1) are consistently oriented, and thus can be checked using the Plücker space algorithm above. The other two pairs of subsets are checked recursively. □

6 High-Dimensional Range Checking

Problem M. Points Outside Intersecting Unit Balls in $\mathbb{R}^3$: Given a set of points and a set of unit balls in $\mathbb{R}^3$, such that every ball contains the origin, is any point contained in any ball?
Best known upper bound: $O(n^{4/3} \log^{4/3} n)$ [3]

This computational problem is closely related to the following open combinatorial problem: What is the worst-case combinatorial complexity of the union of n intersecting unit balls in $\mathbb{R}^3$? The best known bounds are only $O(n^2)$ and $\Omega(n)$. If the complexity of the union is always linear, then Problem M can be solved in $O(n \log n)$ expected time using random sampling techniques [12]. If we allow balls of different sizes, or do not require a common intersection, the union can have complexity $\Omega(n^2)$. The *intersection* of unit balls, on the other hand, always has complexity $O(n)$.

Problem M is a special case of several other harder range checking algorithms. The reductions derive directly from simple geometric transformations. We leave the details as exercises for the reader.

- Unit-spherical range checking in $\mathbb{R}^3$
- Anti-spherical range checking in $\mathbb{R}^3$ (Is *every* point contained in *every* ball?)
- Halfspace range checking in $\mathbb{R}^4$
- Unit anti-spherical range checking in $\mathbb{R}^4$

Problem N. Bichromatic Closest Pair in $\mathbb{R}^3$: Given a set of red and blue points in $\mathbb{R}^3$, find the closest red-blue pair.
Best known upper bound: $O(n^{4/3} \log^{4/3} n)$ [3]

Theorem 13. *Bichromatic closest pair is harder than unit spherical range checking, and unit spherical range checking is almost harder than bichromatic closest pair.*

Proof: We can solve any unit spherical range checking problem by looking for the closest foreign neighbor among the points and the centers of the spheres. If the distance separating the closest point-center pair is less than unity, then the point is in the ball; otherwise, every point is outside every ball. Conversely, we can find the bichromatic closest pair with a parametric search over the space of interpoint distances, using a unit spherical range checking algorithm as an oracle. □

Problem O. Euclidean Minimum Spanning Tree in $\mathbb{R}^3$: Given a set of points in $\mathbb{R}^3$, construct its Euclidean minimum spanning tree.
Best known upper bound: $O(n^{4/3} \log^{4/3} n)$ [3]

Theorem 14. *Euclidean minimum spanning tree is harder than bichromatic closest pair, and bichromatic closest pair is probably harder than Euclidean minimum spanning tree.*

Proof: The first half of the theorem is obvious. The second half follows from a complicated reduction described by Agarwal *et al.* [3, Theorem 5] □

Problem P. Nearest Foreign Neighbors in $\mathbb{R}^3$: Given a set colored points in $\mathbb{R}^3$, find for each point the closest point with a different color.
Best known upper bound: $O(n^{4/3} \log^{4/3} n)$ [1]

Theorem 15 (Yao [28]). *Nearest foreign neighbors is almost harder than Euclidean minimum spanning tree.*

Proof: We can construct the minimum spanning tree using the following algorithm, originally published by Borůvka in 1926 [27]. We start with a forest of n one-vertex trees. In each phase of the algorithm, we find the minimum weight edge leaving each tree, and add it to the evolving forest. After $O(\log n)$ phases, the forest contains only the minimum spanning tree. In the geometric setting, each phase can be easily implemented using a nearest foreign neighbors algorithm. □

Problem Q. Farthest Foreign Pair in $\mathbb{R}^4$: Given a set of red and blue points in $\mathbb{R}^4$, find the farthest red-blue pair.
Best known upper bound: $O(n^{4/3} \log^{O(1)} n)$ [19]

Theorem 16. *Farthest foreign pair is harder than unit anti-spherical range checking, and unit anti-spherical range checking is almost harder than farthest foreign pair.*

Proof: Analogous to the proof of Theorem 13. □

7 Open Problems

We mention one more interesting problem that we have been unable to relate to any of the others.

Problem R. Extreme Points: Given a set of points in $\mathbb{R}^4$, is any point a convex combination of other points? Equivalently, is every point a vertex of the set's convex hull?
Best known upper bound: $O(n^{4/3+\varepsilon})$ [20]

Figure 3 summarizes our results, and suggests a number of open problems. Is there some problem that is easier than both Hopcroft's problem and unit distance detection? Can we better relate the complexities of the problems in Section 5? Is there a single problem that is easier than *all* the problems we have considered?

Ultimately, of course, we would like a proof that all these problems require $\Omega(n^{4/3})$ time in algebraic decision tree model, as we strongly suspect. Unfortunately, proving a lower bound of $\omega(n \log n)$ for *any* decision problem in *any* general model of computation seems to be completely out of reach at present.

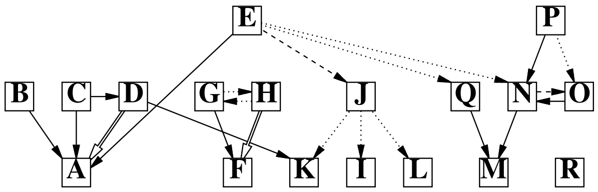


Figure 3. Summary of results. Arrows point from harder to easier problems. Dotted arrows indicate "almost harder". Dashed arrows indicate "probably harder". Outlined arrows indicate results that only hold in the algebraic decision tree model.

References

[1] P. Agarwal and J. Matoušek. Personal communication, reported in [3], 1991.

[2] P. K. Agarwal, B. Aronov, M. Sharir, and S. Suri. Selecting distances in the plane. *Algorithmica*, 9:495–514, 1993.

[3] P. K. Agarwal, H. Edelsbrunner, O. Schwarzkopf, and E. Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. *Discrete Comput. Geom.*, 6:407–422, 1991.

[4] M. Ben-Or. Lower bounds for algebraic computation trees. In *Proc. 15th Annu. ACM Sympos. Theory Comput.*, pages 80–86, 1983.

[5] M. de Berg, M. Overmars, and O. Schwarzkopf. Computing and verifying depth orders. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 138–145, 1992.

[6] S. Bloch, J. Buss, and J. Goldsmith. How hard are n^2-hard problems? *SIGACT News*, 25(2):83–85, 1994.

[7] B. Chazelle. Lower bounds on the complexity of polytope range searching. *J. Amer. Math. Soc.*, 2:637–666, 1989.

[8] B. Chazelle. Lower bounds for off-line range searching. In *Proc. 27th Annu. ACM Sympos. Theory Comput.*, 1995. To appear.

[9] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Diameter, width, closest line pair and parametric searching. *Discrete Comput. Geom.*, 10:183–196, 1993.

[10] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Algorithms for bichromatic line segment problems and polyhedral terrains. *Algorithmica*, 11:116–132, 1994.

[11] B. Chazelle, H. Edelsbrunner, L. J. Guibas, R. Pollack, R. Seidel, M. Sharir, and J. Snoeyink. Counting and cutting cycles of lines and rods in space. In *Proc. 31st Annu. IEEE Sympos. Found. Comput. Sci.*, pages 242–251, 1990.

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

[13] R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM*, 34:200–208, 1987.

[14] J. Erickson. Lower bounds for linear satisfiability problems. In *Proc. 6th Annu. ACM-SIAM Sympos. Discrete Algorithms*, pages 388–395, 1995.

[15] J. Erickson. New lower bounds for Hopcroft's problem. In *Proc. 11th Annu. ACM Sympos. Comput. Geom,*, 1995. To appear.

[16] M. L. Fredman. Lower bounds on the complexity of some optimal data structures. *SIAM J. Comput.*, 10:1–10, 1981.

[17] A. Gajentaan and M. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Comput. Geom. Theory Appl.*, to appear. Appeared previously as: n^2-hard problems in computational geometry. Report RUU-CS-93-15, Dept. Comput. Sci., Utrecht Univ., Utrecht, Netherlands, Apr. 1993.

[18] M. J. Katz and M. Sharir. An expander-based approach to geometric optimization. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 198–207, 1993.

[19] J. Matoušek and O. Schwarzkopf. On ray shooting in convex polytopes. *Discrete Comput. Geom.*, 10(2):215–232, 1993.

[20] J. Matoušek. Linear optimization queries. *J. Algorithms*, 14:432–448, 1993. The results combined with results of O. Schwarzkopf also appear in *Proc. 8th ACM Sympos. Comput. Geom.*, 1992, pages 16–25.

[21] J. Matoušek. Range searching with efficient hierarchical cuttings. *Discrete Comput. Geom.*, 10(2):157–182, 1993.

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

[23] M. Pellegrini. Incidence and nearest-neighbor problems for lines in 3-space. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 130–137, 1992.

[24] F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction*. Springer-Verlag, New York, NY, 1985.

[25] J. M. Steele and A. C. Yao. Lower bounds for algebraic decision trees. *J. Algorithms*, 3:1–8, 1982.

[26] J. Stolfi. *Oriented Projective Geometry: A Framework for Geometric Computations*. Academic Press, 1991.

[27] R. E. Tarjan. *Data Structures and Network Algorithms*, volume 44 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial Applied Mathematics, 1983.

[28] A. C. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM J. Comput.*, 11:721–736, 1982.