Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

AN OPTIMAL ALGORITHM FOR COMPUTING ALL REGULAR LINEAR DEGENERACIES OF POINTSETS IN E^d

Gabriel Robins
Andrew Kahng

December 1990
CSD-900045

An Optimal Algorithm for Computing All Regular Linear Degeneracies of Pointsets in E^d

Gabriel Robins and Andrew Kahng

UCLA Department of Computer Science
Los Angeles, California 90024

Abstract

We give an optimal $O(n^2)$ time algorithm for the problem of finding all maximal equally spaced collinear subsets within a pointset in E^d.

Keywords: Computational geometry, algorithm design, combinatorial problems

1 Introduction

Given a pointset P in E^d, a collinear subset S of P is a subset of P lying on the same line. A maximal collinear subset S of P is a collinear subset of P that is not properly contained in any other collinear subset of P. A maximal collinear equally spaced subset S of P is a maximal collinear subset of P such that the points of S are equally spaced along the line that contains them.

When given a pointset in the plane, it is very natural to ask "what is its largest collinear subset?"

Maximum collinear Subset (MCS) Problem: For n points in E^d, find the largest collinear subset of points.

This type of problem occurs in, e.g., line detection for computer vision, and instances can arise in dimensions greater than two. In practice, bucketing techniques based on the Hough transform [3] [8] [9] or other duality relations in the plane are used. However, for dimension d, such methods usually require effort proportional to $(\frac{1}{\epsilon})^d$ in the desired accuracy ϵ, and are unsuitable for determining exact collinearity as posed here.

The subject of this note is the following very natural variant of MCS:

Maximum Equally-Spaced collinear Subset (MESCS) Problem: For n points in E^d, find the largest collinear, equally spaced subset of points.

Like MCS, the MESCS problem also arises in many practical applications, particularly since *regularity* is in many cases the distinguishing characteristic of "interesting" regions in an image. For example, one might wish to examine infrared ground surveillance bitmaps to find equally spaced collinear "hotspots" (rows of landmines, fenceposts in a region perimeter, etc.) Standard methods for determining "gross" periodicity in data, e.g., spectral methods, do not extend to the present domain.

Often, we would like a roster of all maximal (equally spaced) collinear subsets, which means that the output itself can be of size $O(n^2)$. In other words, we want the complete *order statistics* of the problem on the given input. This note presents an optimal $O(n^2)$ time algorithm for finding the complete order statistics of MESCS for a pointset in arbitrary dimension.

2 Preliminaries

Solving MCS in E^2 can be done in $O(n^2)$ time using an algorithm due to Edelsbrunner [4]. This algorithm actually returns all *maximal proper degeneracies* for a pointset in E^d, i.e., all maximal subsets of points lying in overdetermined k-flats, and runs in time $O(n^d)$. The exponential dependence on dimension is expensive, except for very small d, and we would like to avoid this cost, particularly since we only wish to find maximal collinear subsets. Furthermore, it is not clear how the method of [4] extends to satisfy the "equally spaced" constraint. Indeed, the notion of equally spaced points is difficult to define elegantly in k-flats with $k > 1$.

The work of Edelsbrunner and Guibas [6] also implies an $O(n^2)$ time algorithm for MCS in two dimensions, but this method does not generalize to higher dimensions, nor does it seem applicable to the MESCS problem. Avis and Doskas [1] outline methods that given a set of segments or polyhedra in E^d determine whether a single hyperplane intersects *all* of these objects, but their algorithms are exponential in d, and we are interested rather in the *maximum* number of "stabbed" objects.

For both MCS and MESCS, a lower bound may be established by reduction from the Element Uniqueness problem (i.e., determining whether a given set of numbers contains duplicates), which is known to require $\Omega(n \log n)$ time [7].

Theorem One: MCS and MESCS both require at least $\Omega(n \log n)$ time.

Proof: Given an instance of EU $S = \{x_1, \ldots, x_n\}$, we reduce it to the two-

dimensional MCS problem as follows. Transform each element x_i into (i, x_i^2). The resulting instance of MCS has a collinear subset with cardinality 3 or greater *iff* S contains duplicates, and the $\Omega(n \log n)$ lower bound for EU translates into one for MCS. Similarly, EU is reduced to MESCS as follows. Given an instance of EU $S = \{x_1, \ldots, x_n\}$, transform each element x_i into $y_i = (1 + \epsilon/2^{x_i}) \cdot x_i$, and form the multi-set $S' = \{y_1, \ldots, y_n, y_1, \ldots, y_n\}$. Clearly this transformation destroys any arithmetic progressions that may be present in the original data, except for the trivial ones that are induced by duplicate elements; that is, S' contains 3 or more "equally-spaced" points (with zero spacing) *iff* S contains duplicate elements. The $\Omega(n \log n)$ lower bound for EU thus holds for MESCS in E^d, $d > 0$.

□

We close this section with a brief discussion of simple methods for the MCS problem in all dimensions. Clearly MCS can be solved naively in time $O(n^3)$ by iterating through all $\binom{n}{2}$ lines induced by the pointset, and for each line determining which of the n points lie on that line. The following algorithm solves MCS in $O(n^2 \log n)$ time: for each of the $\binom{n}{2}$ lines induced by the pointset, represent that line by a unique pair of parameters (e.g., "slope/y-intercept", or "angle-of-normal/distance-to-origin", etc.) Next, sort the lines using these two parameters as primary and secondary sort keys, respectively: this requires $O(n^2 \log n)$ time. Pass through the list and check for duplicate representations; these duplicates represent sets of collinear points, with a set of k collinear points being represented by $\binom{k}{2}$ elements on this list. This solves the MCS problem within time $O(n^2 \log n)$, and we actually may output the *complete* order statistics, i.e., *all* lines induced by the original pointset, along with the number of points lying on each line, within this asymptotic time bound.

3 The Maximum Equally Spaced collinear Subset Problem

This section develops a succession of progressively improved methods for solving the MESCS problem. We shall first solve this problem in one dimension.

Begin by sorting the input values. Next, for every pair of numbers x_i and x_j in the sorted list, we try to determine the longest arithmetic progression starting with x_i and x_j, $i < j$, and having a successive element difference of x_j - x_i; subsequent elements in this arithmetic progression may be searched for in the sorted list using binary search at logarithmic cost per element. In order to avoid duplicate searches in the future, whenever we obtain another element in an arithmetic progression in this way, we update an auxiliary two-dimensional matrix, so that a possible future arithmetic progressions (having the same tail

elements as the current one) need not be "chased" again to their end. This ensures that each pair of elements needs only to be examined once, bringing the total time for the entire algorithm to $O(n^2 \log n)$. Note that within the same asymptotic time we computed not only the longest arithmetic progression in the data, but *all* other maximal ones as well.

Our first approach for solving MESCS in arbitrary dimension computes all the lines determined by the pointset along with the points that fall of each of these lines; the one dimensional algorithm is then used to compute the order statistics for each line. Intuitively, the approach succeeds because when more points lie on each line, we must have fewer lines: when we decompose a higher dimensional instance into a collection of one-dimensional instances, the inverse relationship between the size of these instances and their number will thus keep the overall time complexity relatively low. The formal analysis is as follows.

Given a pointset, perturb all the points by a tiny amount until no three are collinear; this is always possible. Now return the points to their original positions one at a time while observing the quantity

$$\Phi = \sum_{L \in \{lines\}} \#points(L)$$

Fact: Φ takes on its maximum value $n \cdot (n-1)$ when no three points are collinear.

Proof: Moving a perturbed point to its original location on a line that has already $k > 1$ points on it will decrease Φ by at least k. □

In returning the perturbed points back to their original locations, by the time we obtain k collinear points on a given line, Φ has been reduced by at least $\Omega(k^2)$. Thus, the number of lines (induced by the original pointset) that contain exactly k points is at most $O(\Phi/k^2)$, and in particular this quantity is never greater than $O(n^2/k^2)$. Processing a line that contains exactly k points, using the one-dimensional algorithm above, requires time $O(k^2 \log k)$; the total time required to process at most $O(n^2/k^2)$ such lines is bounded by $O(n^2 \log k)$. Summing this over all values of k (as k ranges between 1 and n) yields a grand total of $O(n^3 \log k)$ time to solve MESCS.

Observe that for all k between $n/2^i$ and $n/2^{i+1}$, at most $O(n^2/(n/2^{i+1})^2)$ lines may contain k points, and each may be processed using the one-dimensional algorithm in time at most $O((n/2^i)^2 \log(n/2^i))$. Therefore the time to process all lines containing between $n/2^i$ and $n/2^{i+1}$ points each is

$$O(n^2/(n/2^{i+1})^2) \cdot O((n/2^i)^2 \log(n/2^i)) = O(n^2 \log n)$$

Summing over i ranging between 0 and $\log n$ yields a total of $O(n^2 \log^2 n)$

time to solve MESCS in E^2.

Finally, note that each of the m lines l_i in the configuration, each containing k_i points of P respectively, will reduce Φ by $O(k_i^2)$. However, Φ remains positive, yielding the bound

$$\sum_{i=1}^{m} k_i^2 = O(n^2)$$

Thus the time to process all of the m lines using the one-dimensional algorithm is given by

$$\sum_{i=1}^{m} k_i^2 \log k_i \leq \sum_{i=1}^{m} k_i^2 \log n = \log n \cdot \sum_{i=1}^{m} k_i^2 = O(n^2 \log n)$$

Thus the approach of processing all lines of the configuration separately will afford a solution to MESCS in arbitrary dimension within time $O(n^2 \log n)$. The next paragraphs describe how to reduce the complexity of our solution to $O(n^2)$ in all dimensions.

Let us reconsider the one-dimensional MESCS problem and the special variant which looks for an equally spaced *triple* of points, i.e., an arithmetic progression of length three. (We can show that this problem, as well as the problem of determining whether a given pointset contains a collinear triplet, also has an $\Omega(n \log n)$ lower bound, using the same reductions as in the proof of Theorem One. Interestingly, for neither of these apparently much simpler decision problems is an $o(n^2)$ time algorithm known [5].

To find all equally spaced triples, first sort the input values. Now assume that the leftmost point A of each triple is X_i, and advance two pointers B and C beginning at x_{i+1} and x_{i+2} respectively. If $x_{i+1} - x_i > x_{i+2} - x_{i+1}$, we advance pointer C, otherwise we advance pointer B. Whenever the two differences are equal, we record the corresponding equally spaced triple (A,B,C). Clearly this process will determine in linear time all equally spaced triples with x_i as the first component of the triple; iterating over all values of i will therefore report all equally spaced triples in the data within time $O(n^2)$. It is possible to construct inputs which have a quadratic number of such triples (e.g., all points equally spaced), so this method is optimal.

Our final reduction in the time complexity of one-dimensional MESCS is as follows. Detect all equally spaced triples of points (using $O(n^2)$ time), and then overlap them in order to determine all maximal equally spaced chains. In other words, we construct a directed graph where for each reported equally spaced triple x_i, x_j, and x_k we create the nodes $< i, j >$ and $< j, k >$ and the directed

edge ($< i, j >, < j, k >$). Each node in this graph has indegree and outdegree of at most one, so the edge set and vertex set are both of size $O(n^2)$. A topological sort of this directed graph yields all maximal equally spaced subsets in $O(n^2)$ time.

To solve MESCS in higher dimensions, we sort the pointset by the first coordinate only; i.e., we project onto the x_1 axis. Without loss of generality, we can assume that no two points have the same x_1 coordinate (we can always rotate the pointset by a tiny angle to make the x_1 coordinates unique). We then proceed to solve the 1-dimensional MESCS problem for the sorted, projected pointset. Equally spaced triplets will correspond to equally spaced triplets in the projection. However, some equally spaced triplets in the projection will not correspond to actual equally spaced triplets; this can be easily checked in constant time per triplet for any fixed dimension. Since the number of equally spaced triplets is bounded by $\binom{n}{2}$ in all dimensions, our algorithm will run in time $O(n^2)$ for any fixed dimension.

4 Conclusion

We gave optimal algorithms and proved lower bounds for computing all order statistics of cardinalities of equally-spaced collinear subsets within a pointset in arbitrary dimensions.

5 Acknowledgement

We thank Alfredo Inselberg for bringing several references to our attention.

References

[1] D. Avis and M. Doskas, "Algorithms for High Dimensional Stabbing Problems", *Discrete Applied Mathematics* 27(1990), pp. 39-48.

[2] R. Cole, J. S. Slowe, W. L. Steigers, and E. Szemeredi, "An Optimal-Time Algorithm for Slope Selection", *Siam J. Computing* 18 (4)(1989), pp. 792-810.

[3] R. Duda, and P. Hart, "Use of the Hough Transform to Detect Lines and Curves in Pictures", *Communications of the ACM* 15 (1)(1972), pp. 11-15.

[4] H. Edelsbrunner, *Algorithms in Computational Geometry*, Springer-Verlag, Berlin, 1987, pp. 278-282.

[5] H. Edelsbrunner, J. O'Rourke, and R. Seidel, "Constructing Arrangements of Lines and Hyperplanes With Applications", *SIAM J. Computing* 15(2)(1986), pp. 341-363.

[6] H. Edelsbrunner and L. J. Guibas, "Topologically Sweeping an Arrangement", *Proc. ACM Symposium on Theory of Computing* , 1986, pp. 389-403.

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

[8] T. Risse, "Hough Transform for Line Recognition: Complexity of Evidence Accumulation and Cluster Detection", *Computer Vision* 46(1989), pp. 327-345.

[9] D. Ben-Tzvi and M. B. Sandler, "A Combinatorial Hough Transform", *Pattern Recognition Letters* 11(1990), pp. 167-174.

Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

AN OPTIMAL ALGORITHM FOR COMPUTING ALL REGULAR LINEAR DEGENERACIES OF POINTSETS IN E^d

Gabriel Robins
Andrew Kahng

December 1990
CSD-900045

An Optimal Algorithm for Computing All Regular Linear Degeneracies of Pointsets in E^d

Gabriel Robins and Andrew Kahng

UCLA Department of Computer Science
Los Angeles, California 90024

Abstract

We give an optimal $O(n^2)$ time algorithm for the problem of finding all maximal equally spaced collinear subsets within a pointset in E^d.

Keywords: Computational geometry, algorithm design, combinatorial problems

1 Introduction

Given a pointset P in E^d, a collinear subset S of P is a subset of P lying on the same line. A maximal collinear subset S of P is a collinear subset of P that is not properly contained in any other collinear subset of P. A maximal collinear equally spaced subset S of P is a maximal collinear subset of P such that the points of S are equally spaced along the line that contains them.

When given a pointset in the plane, it is very natural to ask "what is its largest collinear subset?"

Maximum collinear Subset (MCS) Problem: For n points in E^d, find the largest collinear subset of points.

This type of problem occurs in, e.g., line detection for computer vision, and instances can arise in dimensions greater than two. In practice, bucketing techniques based on the Hough transform [3] [8] [9] or other duality relations in the plane are used. However, for dimension d, such methods usually require effort proportional to $(\frac{1}{\epsilon})^d$ in the desired accuracy ϵ, and are unsuitable for determining exact collinearity as posed here.

The subject of this note is the following very natural variant of MCS:

Maximum Equally-Spaced collinear Subset (MESCS) Problem: For n points in E^d, find the largest collinear, equally spaced subset of points.

Like MCS, the MESCS problem also arises in many practical applications, particularly since *regularity* is in many cases the distinguishing characteristic of "interesting" regions in an image. For example, one might wish to examine infrared ground surveillance bitmaps to find equally spaced collinear "hotspots" (rows of landmines, fenceposts in a region perimeter, etc.) Standard methods for determining "gross" periodicity in data, e.g., spectral methods, do not extend to the present domain.

Often, we would like a roster of all maximal (equally spaced) collinear subsets, which means that the output itself can be of size $O(n^2)$. In other words, we want the complete *order statistics* of the problem on the given input. This note presents an optimal $O(n^2)$ time algorithm for finding the complete order statistics of MESCS for a pointset in arbitrary dimension.

2 Preliminaries

Solving MCS in E^2 can be done in $O(n^2)$ time using an algorithm due to Edelsbrunner [4]. This algorithm actually returns all *maximal proper degeneracies* for a pointset in E^d, i.e., all maximal subsets of points lying in overdetermined k-flats, and runs in time $O(n^d)$. The exponential dependence on dimension is expensive, except for very small d, and we would like to avoid this cost, particularly since we only wish to find maximal collinear subsets. Furthermore, it is not clear how the method of [4] extends to satisfy the "equally spaced" constraint. Indeed, the notion of equally spaced points is difficult to define elegantly in k-flats with $k > 1$.

The work of Edelsbrunner and Guibas [6] also implies an $O(n^2)$ time algorithm for MCS in two dimensions, but this method does not generalize to higher dimensions, nor does it seem applicable to the MESCS problem. Avis and Doskas [1] outline methods that given a set of segments or polyhedra in E^d determine whether a single hyperplane intersects *all* of these objects, but their algorithms are exponential in d, and we are interested rather in the *maximum* number of "stabbed" objects.

For both MCS and MESCS, a lower bound may be established by reduction from the Element Uniqueness problem (i.e., determining whether a given set of numbers contains duplicates), which is known to require $\Omega(n \log n)$ time [7].

Theorem One: MCS and MESCS both require at least $\Omega(n \log n)$ time.

Proof: Given an instance of EU $S = \{x_1, \ldots, x_n\}$, we reduce it to the two-

dimensional MCS problem as follows. Transform each element x_i into (i, x_i^2). The resulting instance of MCS has a collinear subset with cardinality 3 or greater *iff* S contains duplicates, and the $\Omega(n \log n)$ lower bound for EU translates into one for MCS. Similarly, EU is reduced to MESCS as follows. Given an instance of EU $S = \{x_1, \ldots, x_n\}$, transform each element x_i into $y_i = (1 + \epsilon/2^{x_i}) \cdot x_i$, and form the multi-set $S' = \{y_1, \ldots, y_n, y_1, \ldots, y_n\}$. Clearly this transformation destroys any arithmetic progressions that may be present in the original data, except for the trivial ones that are induced by duplicate elements; that is, S' contains 3 or more "equally-spaced" points (with zero spacing) *iff* S contains duplicate elements. The $\Omega(n \log n)$ lower bound for EU thus holds for MESCS in E^d, $d > 0$. □

We close this section with a brief discussion of simple methods for the MCS problem in all dimensions. Clearly MCS can be solved naively in time $O(n^3)$ by iterating through all $\binom{n}{2}$ lines induced by the pointset, and for each line determining which of the n points lie on that line. The following algorithm solves MCS in $O(n^2 \log n)$ time: for each of the $\binom{n}{2}$ lines induced by the pointset, represent that line by a unique pair of parameters (e.g., "slope/y-intercept", or "angle-of-normal/distance-to-origin", etc.) Next, sort the lines using these two parameters as primary and secondary sort keys, respectively: this requires $O(n^2 \log n)$ time. Pass through the list and check for duplicate representations; these duplicates represent sets of collinear points, with a set of k collinear points being represented by $\binom{k}{2}$ elements on this list. This solves the MCS problem within time $O(n^2 \log n)$, and we actually may output the *complete* order statistics, i.e., *all* lines induced by the original pointset, along with the number of points lying on each line, within this asymptotic time bound.

3 The Maximum Equally Spaced collinear Subset Problem

This section develops a succession of progressively improved methods for solving the MESCS problem. We shall first solve this problem in one dimension.

Begin by sorting the input values. Next, for every pair of numbers x_i and x_j in the sorted list, we try to determine the longest arithmetic progression starting with x_i and x_j, $i < j$, and having a successive element difference of x_j - x_i; subsequent elements in this arithmetic progression may be searched for in the sorted list using binary search at logarithmic cost per element. In order to avoid duplicate searches in the future, whenever we obtain another element in an arithmetic progression in this way, we update an auxiliary two-dimensional matrix, so that a possible future arithmetic progressions (having the same tail

elements as the current one) need not be "chased" again to their end. This ensures that each pair of elements needs only to be examined once, bringing the total time for the entire algorithm to $O(n^2 \log n)$. Note that within the same asymptotic time we computed not only the longest arithmetic progression in the data, but *all* other maximal ones as well.

Our first approach for solving MESCS in arbitrary dimension computes all the lines determined by the pointset along with the points that fall of each of these lines; the one dimensional algorithm is then used to compute the order statistics for each line. Intuitively, the approach succeeds because when more points lie on each line, we must have fewer lines: when we decompose a higher dimensional instance into a collection of one-dimensional instances, the inverse relationship between the size of these instances and their number will thus keep the overall time complexity relatively low. The formal analysis is as follows.

Given a pointset, perturb all the points by a tiny amount until no three are collinear; this is always possible. Now return the points to their original positions one at a time while observing the quantity

$$\Phi = \sum_{L \in \{lines\}} \#points(L)$$

Fact: Φ takes on its maximum value $n \cdot (n-1)$ when no three points are collinear.

Proof: Moving a perturbed point to its original location on a line that has already $k > 1$ points on it will decrease Φ by at least k. □

In returning the perturbed points back to their original locations, by the time we obtain k collinear points on a given line, Φ has been reduced by at least $\Omega(k^2)$. Thus, the number of lines (induced by the original pointset) that contain exactly k points is at most $O(\Phi/k^2)$, and in particular this quantity is never greater than $O(n^2/k^2)$. Processing a line that contains exactly k points, using the one-dimensional algorithm above, requires time $O(k^2 \log k)$; the total time required to process at most $O(n^2/k^2)$ such lines is bounded by $O(n^2 \log k)$. Summing this over all values of k (as k ranges between 1 and n) yields a grand total of $O(n^3 \log k)$ time to solve MESCS.

Observe that for all k between $n/2^i$ and $n/2^{i+1}$, at most $O(n^2/(n/2^{i+1})^2)$ lines may contain k points, and each may be processed using the one-dimensional algorithm in time at most $O((n/2^i)^2 \log(n/2^i))$. Therefore the time to process all lines containing between $n/2^i$ and $n/2^{i+1}$ points each is

$$O(n^2/(n/2^{i+1})^2) \cdot O((n/2^i)^2 \log(n/2^i)) = O(n^2 \log n)$$

Summing over i ranging between 0 and $\log n$ yields a total of $O(n^2 \log^2 n)$

time to solve MESCS in E^2.

Finally, note that each of the m lines l_i in the configuration, each containing k_i points of P respectively, will reduce Φ by $O(k_i^2)$. However, Φ remains positive, yielding the bound

$$\sum_{i=1}^{m} k_i^2 = O(n^2)$$

Thus the time to process all of the m lines using the one-dimensional algorithm is given by

$$\sum_{i=1}^{m} k_i^2 \log k_i \leq \sum_{i=1}^{m} k_i^2 \log n = \log n \cdot \sum_{i=1}^{m} k_i^2 = O(n^2 \log n)$$

Thus the approach of processing all lines of the configuration separately will afford a solution to MESCS in arbitrary dimension within time $O(n^2 \log n)$. The next paragraphs describe how to reduce the complexity of our solution to $O(n^2)$ in all dimensions.

Let us reconsider the one-dimensional MESCS problem and the special variant which looks for an equally spaced *triple* of points, i.e., an arithmetic progression of length three. (We can show that this problem, as well as the problem of determining whether a given pointset contains a collinear triplet, also has an $\Omega(n \log n)$ lower bound, using the same reductions as in the proof of Theorem One. Interestingly, for neither of these apparently much simpler decision problems is an $o(n^2)$ time algorithm known [5].

To find all equally spaced triples, first sort the input values. Now assume that the leftmost point A of each triple is X_i, and advance two pointers B and C beginning at x_{i+1} and x_{i+2} respectively. If $x_{i+1} - x_i > x_{i+2} - x_{i+1}$, we advance pointer C, otherwise we advance pointer B. Whenever the two differences are equal, we record the corresponding equally spaced triple (A,B,C). Clearly this process will determine in linear time all equally spaced triples with x_i as the first component of the triple; iterating over all values of i will therefore report all equally spaced triples in the data within time $O(n^2)$. It is possible to construct inputs which have a quadratic number of such triples (e.g., all points equally spaced), so this method is optimal.

Our final reduction in the time complexity of one-dimensional MESCS is as follows. Detect all equally spaced triples of points (using $O(n^2)$ time), and then overlap them in order to determine all maximal equally spaced chains. In other words, we construct a directed graph where for each reported equally spaced triple x_i, x_j, and x_k we create the nodes $< i, j >$ and $< j, k >$ and the directed

edge ($< i, j >, < j, k >$). Each node in this graph has indegree and outdegree of at most one, so the edge set and vertex set are both of size $O(n^2)$. A topological sort of this directed graph yields all maximal equally spaced subsets in $O(n^2)$ time.

To solve MESCS in higher dimensions, we sort the pointset by the first coordinate only; i.e., we project onto the x_1 axis. Without loss of generality, we can assume that no two points have the same x_1 coordinate (we can always rotate the pointset by a tiny angle to make the x_1 coordinates unique). We then proceed to solve the 1-dimensional MESCS problem for the sorted, projected pointset. Equally spaced triplets will correspond to equally spaced triplets in the projection. However, some equally spaced triplets in the projection will not correspond to actual equally spaced triplets; this can be easily checked in constant time per triplet for any fixed dimension. Since the number of equally spaced triplets is bounded by $\binom{n}{2}$ in all dimensions, our algorithm will run in time $O(n^2)$ for any fixed dimension.

4 Conclusion

We gave optimal algorithms and proved lower bounds for computing all order statistics of cardinalities of equally-spaced collinear subsets within a pointset in arbitrary dimensions.

5 Acknowledgement

We thank Alfredo Inselberg for bringing several references to our attention.

References

[1] D. Avis and M. Doskas, "Algorithms for High Dimensional Stabbing Problems", *Discrete Applied Mathematics* 27(1990), pp. 39-48.

[2] R. Cole, J. S. Slowe, W. L. Steigers, and E. Szemeredi, "An Optimal-Time Algorithm for Slope Selection", *Siam J. Computing* 18 (4)(1989), pp. 792-810.

[3] R. Duda, and P. Hart, "Use of the Hough Transform to Detect Lines and Curves in Pictures", *Communications of the ACM* 15 (1)(1972), pp. 11-15.

[4] H. Edelsbrunner, *Algorithms in Computational Geometry*, Springer-Verlag, Berlin, 1987, pp. 278-282.

[5] H. Edelsbrunner, J. O'Rourke, and R. Seidel, "Constructing Arrangements of Lines and Hyperplanes With Applications", *SIAM J. Computing* 15(2)(1986), pp. 341-363.

[6] H. Edelsbrunner and L. J. Guibas, "Topologically Sweeping an Arrangement", *Proc. ACM Symposium on Theory of Computing* , 1986, pp. 389-403.

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

[8] T. Risse, "Hough Transform for Line Recognition: Complexity of Evidence Accumulation and Cluster Detection", *Computer Vision* 46(1989), pp. 327-345.

[9] D. Ben-Tzvi and M. B. Sandler, "A Combinatorial Hough Transform", *Pattern Recognition Letters* 11(1990), pp. 167-174.