Finding squares and rectangles in sets of points

M.J. van Kreveld and M.T. de Berg

RUU-CS-89-10
April 1989

Finding squares and rectangles in sets of points

M.J. van Kreveld and M.T. de Berg

RUU-CS-89-10
April 1989

Rijksuniversiteit Utrecht

Vakgroep informatica

Padualaan 14 3584 CH Utrecht
Corr. adres: Postbus 80.089, 3508 TB Utrecht
Telefoon 030-531454
The Netherlands

Finding Squares and Rectangles in Sets of Points

Marc J. van Kreveld Mark T. de Berg

Technical Report RUU-CS-89-10
April 1989

Department of Computer Science
University of Utrecht
P.O.Box 80.089
3508 TB Utrecht
the Netherlands

Finding Squares and Rectangles in Sets of Points

Marc J. van Kreveld Mark T. de Berg*

Dept. of Computer Science, University of Utrecht,
P.O.Box 80.089, 3508 TB Utrecht, the Netherlands.

Abstract

The following problem is studied: Given a set S of n points in the plane, does it contain a subset of four points that form the vertices of a square or rectangle. Both the axis-parallel case and the arbitrarily oriented case are studied. We also investigate extensions to the d-dimensional case. Algorithms are obtained that run in $O(n^{1+1/d} \log n)$ time for axis-parallel squares and $O(n^{2-1/d})$ time for axis-parallel rectangles. For arbitrarily oriented squares the time bounds are $O(n^2 \log n)$, $O(n^3)$ and $O(n^{d-1/2}\beta(n))$ for $d = 2$, $d = 3$ and $d \geq 4$, respectively (where $\beta(n)$ is related to the inverse of Ackermann's function), whereas the algorithm for arbitrarily oriented rectangles takes time $O(n^d \log n)$. Also combinatorial results on the maximal number of squares and rectangles any point set can contain are given.

1 Introduction

In this paper we study the problem of determining whether a set of points contains some special configurations of points. Recognizing such 'degeneracies' can be important in cases where they cause problems. In particular we will search for vertices of squares or rectangles. The following problems are considered.

Problem 1 *Given a set S of points in the plane, determine whether S contains a subset of four points that form the vertices of an axis-parallel square.*

Problem 2 *Given a set S of points in the plane, determine whether S contains a subset of four points that form the vertices of an axis-parallel rectangle.*

*The work of this author is supported by the Dutch Organisation for Scientific Research (N.W.O.).

A trivial method, exploiting the fact that two opposite vertices define the square or rectangle, yields an $O(n^2 \log n)$ time bound for the problems. We present algorithms that solve the problems in time $O(n\sqrt{n}\log n)$ and $O(n\sqrt{n})$, respectively. We will show that the rectangle problem is equivalent to the problem of determining whether a bipartite graph contains a $K_{2,2}$ subgraph, and obtain an $O(|E|\sqrt{|E|})$ time solution.

The results extend to d-dimensional space, and we obtain an $O(n^{1+1/d}\log n)$ time algorithm for the square problem, and an $O(n^{2-1/d})$ time algorithm for the rectangle problem. Note that the square problem becomes easier to solve, while the rectangle problem becomes harder to solve for higher dimensions. Both algorithms can be adapted to solve the 'report all squares/rectangles' problem. Then the output size is added as a linear term in the time bound.

The combinatorial side of the problems is studied as well: What is the maximal number of subsets of a set of n points that are the vertices of a square or rectangle. We prove tight bounds of $\Theta(n^{1+1/d})$ and $\Theta(n^2)$, respectively.

We also study the following related problems.

Problem 3 *Given a set S of points in the plane, determine whether S contains a subset of four points that form the vertices of an arbitrarily oriented square.*

Problem 4 *Given a set S of points in the plane, determine whether S contains a subset of four points that form the vertices of an arbitrarily oriented rectangle.*

Note that in these problems the restiction that the object is axis-parallel has been dropped.

In the d-dimensional case we find rectangles $O(n^d \log n)$ time, whereas the square problem can be solved in time $O(n^2 \log n)$ $(d = 2)$, $O(n^3)$ $(d = 3)$, or $O(n^{d-1/2}\beta(n))$ $(d \geq 4)$, where $\beta(n)$ is related to the extremely slowly growing inverse of Ackermann's function. Again, our methods can easily be adapted to report all squares and rectangles. Also, we show that the maximal number of squares in the plane is $\Theta(n^2)$. Tight bounds on the maximal number of rectangles in the plane are not known: the trivial lower bound is $\Omega(n^2)$, whereas we prove an upper bound of $O(n^2\sqrt{n})$.

This paper is organized as follows.

In section 2, the axis-parallel square problem is considered. We obtain combinatorial results on the maximal number of squares any planar set of points can contain, and we give an algorithm for determining whether a planar set of points contains the vertices of a square. Both results are extended to higher dimensions.

In section 3, the axis-parallel rectangle problem is considered, and all problems of section 2 are now studied for rectangles instead of squares. Additionally, we show equivalence of the rectangle determination problem to finding a $K_{2,2}$ subgraph in a bipartite graph.

Section 4 deals with arbitrarily oriented squares and rectangles in sets of points. We give combinatorial results in the planar case, and algorithms in the arbitrarily dimensional case.

In section 5 we conclude the paper by mentioning some directions for further research.

2 Axis-parallel squares

When we restrict ourselves to axis-parallel shapes, we can make use of the fact that these shapes contain points with a number of coordinates equal-valued. We use this fact in this section and the next for recognizing these shapes. In this section we will study the problem of recognizing squares in the plane and more-dimensional space. We begin with a combinatorial result.

Theorem 1 *For a set S of n points in d-dimensional space, the maximal number of subsets of 2^d points that form the vertices of an axis-parallel hypercube is $\Theta(n^{1+1/d})$.*

Proof: We first consider the planar case. To prove the lower bound, place the points of S on a grid G of size $\lfloor\sqrt{n}\rfloor \times \lfloor\sqrt{n}\rfloor$. Let p be a point in the lower left quadrant of G. For each point q on the diagonal to the right above p there is a square with p as lower left vertex and q as top right vertex. Hence, we find at least $\lfloor\sqrt{n}/2\rfloor - 1$ squares, which are all different. As there are $\lfloor\sqrt{n}\rfloor^2/4$ places for p in the lower left quadrant, there are $\Omega(n\sqrt{n})$ axis-parallel squares with all vertices on the grid.

To prove the upper bound, partition S in a number of subsets $S_1, \ldots, S_k$, such that two points p and q are in the same subset if and only if they have equal first coordinate. If some subset S_i contains no more than $\sqrt{n}$ points, then every point of S_i can participate in at most $\sqrt{n} - 1$ squares. Consequently, there are at most $n\sqrt{n}$ squares in total with at least one vertex in any subset of size at most $\sqrt{n}$. Next, consider the subsets with more than $\sqrt{n}$ points. There are less than $\sqrt{n}$ such subsets. If the points in these subsets are partitioned in new subsets of points with equal second coordinate, then these new subsets contain less than $\sqrt{n}$ points, and hence these subsets contain less than $n\sqrt{n}$ squares.

The d-dimensional case follows in a similar way. To prove the lower bound, use a grid of size $\lfloor n^{1/d}\rfloor \times \cdots \times \lfloor n^{1/d}\rfloor$. For the upper bound, partition S in a number of subsets with all but the last coordinate equal. The subsets of size at most $n^{1/d}$ cannot contribute to more than $n^{1+1/d}$ hypercubes, and the points in larger subsets can be partitioned in subsets with only some other coordinate different. Details are left to the reader. $\Box$

Let S be a planar set of n points. The main idea to solve the two-dimensional square problem is to partition S in a number of subsets, such that all points with equal first coordinate are in the same subset. Each pair of points in such a subset determines two possible squares, such that the pair forms the vertices of one vertical edge of the square, and it remains to determine if S contains the two vertices of an opposite edge, either to the left or to the right of the first edge. One could simply search for these four points. Unfortunately, there may be as many as $\Omega(n)$ points with equal first coordinate, hence, there can be $\Omega(n^2)$ pairs in one subset, resulting in a $O(n^2 \log n)$ time algorithm ($O(\log n)$ time for searching). To improve upon this we will show that we can avoid having to test all pairs in large subsets by observing that we could have made subsets of points with equal second coordinate equally well.

Lemma 1 *Given a sets S of n (distinct) points in the plane, then there exists an axis-parallel line that contains at least one and at most $\sqrt{n}$ points of S.*

Proof: Partition S in non-empty subsets $S_1, \ldots, S_k$ of points with equal first coordinate. If any subset contains at most $\sqrt{n}$ points, then we are done, for we can take a vertical line containing this subset. If all subsets contain more than $\sqrt{n}$ points, then obviously there are at most $\sqrt{n}$ subsets. Since two points in the same subset cannot have equal second coordinate (the points are distinct), every horizontal line will contain at most one point of each subset. Thus in this case, any horizontal line containing a point of S will do. □

According to the proof of the above lemma, we can partition the set S of points in a number of subsets of all points of S with equal first coordinate, and a number of subsets of the remaining points of S with equal second coordinate, such that no subset contains more than $\sqrt{n}$ points of S.

Square (S)

1. Build a search tree T on the set S of points, using lexicographic order on the points.
2. Partition S into subsets $S_1^{(1)}, \ldots, S_{k_1}^{(1)}, S_1^{(2)}, \ldots, S_{k_2}^{(2)}$ in the following way. Two points of S are in the same subset $S_i^{(1)}$ ($1 \leq i \leq k_1$) if and only if they have equal first coordinate, and $S_i^{(1)}$ does not contain more than $\sqrt{n}$ points. Two points of S are in the same subset $S_j^{(2)}$ ($1 \leq j \leq k_2$) if and only if they have equal second coordinate, and neither of them is in some subset $S_i^{(1)}$.
3. For every subset $S_i^{(j)}$ ($1 \leq j \leq 2$, $1 \leq i \leq k_j$), and for every pair p, q in $S_i^{(j)}$, search in T whether the other two vertices of any of the two squares defined by p and q are also in S. If so, answer yes, otherwise, answer no.

This leads to the following result.

Theorem 2 *Given a set S of n points in the plane, it can be decided in time $O(n\sqrt{n}\log n)$ and $O(n)$ space whether S contains a subset of four points that form the vertices of an axis-parallel square.*

Proof: The correctness of the algorithm given above is clear, since any square will have at least one pair of vertices of an edge in one subset, and this pair will certainly be tested.

Step 1 and 2 of the algorithm clearly take $O(n \log n)$ time.

According to lemma 1, the partitioning of S in subsets as given by the algorithm results in subsets with at most $\sqrt{n}$ points. Thus the total time spent by step 3 is

$$O(\sum_{i=1}^{k_1} |S_i^{(1)}|^2 \cdot \log n + \sum_{i=1}^{k_2} |S_i^{(2)}|^2 \cdot \log n) = O(\sqrt{n} \log n (\sum_{i=1}^{k_1} |S_i^{(1)}| + \sum_{i=1}^{k_2} |S_i^{(2)}|)) =$$

$$O(n\sqrt{n} \log n).$$

□

Remark: In fact, we can solve the above problem in time $O(n\sqrt{n \log n})$ in the following way. Partition S on equal first coordinate, resulting in small subsets with at most $\sqrt{n/\log n}$ points, and large subsets with more points. Treat the small subsets as above, and test all pairs of large subsets by a simultaneous walk at two places in both (ordered) large subsets. Unfortunately this approach does not generalize to higher dimensions.

Next we consider the more-dimensional case of the problem: Given a set S of n points in d-dimensional space, does S contain a subset of 2^d points that are the vertices of an axis-parallel (hyper-) cube. Note that two points with all but the last coordinate equal-valued determine 2^{d-1} possible cubes, and every candidate cube thus found can be tested by searching in S for the $2^d - 2$ points that would be the other vertices of the cube.

Lemma 2 *Given a set S of n (distinct) points in d-dimensional space, then there exists an axis-parallel line that contains at least one and at most $n^{1/d}$ points of S.*

Proof: With induction on d. For $d = 2$ the proof is given in lemma 1.

Let $d > 2$. Partition S in subsets $S_1, \ldots, S_k$ such that two points are in the same subset S_i if and only if they have equal last coordinate. If there are at most $n^{1/d}$ subsets then any line parallel to the last coordinate axis containing at least one point of S will do. Otherwise, there must be a subset S_i with at most $n^{1-1/d}$ points. From the induction hypothesis, in this subset of points in $(d-1)$-dimensional space there is a line parallel to one of the first $d-1$ coordinate axes with at least one and at most $(n^{1-1/d})^{1/(d-1)} = n^{1/d}$ points of S. Obviously, this line contains no points of other subsets. □

The following algorithm solves the cube problem by partitioning the set of points S in subsets of points with all but one coordinate equal, and which contain at most $n^{1/d}$ points.

Cube (S, d)

1. Build a search tree T on the set S of points, using lexicographic order on the points.

2. Partition S into subsets $S_1^{(1)}, \ldots, S_{k_1}^{(1)}, S_1^{(2)}, \ldots, S_{k_2}^{(2)}, \ldots\ldots, S_1^{(d)}, \ldots, S_{k_d}^{(d)}$ in the following way. Two points of S are in the same subset $S_i^{(j)}$ ($1 \leq j \leq d$, $1 \leq i \leq k_j$) if and only if they have all coordinates but the j^{th} equal, $S_i^{(j)}$ does not contain more than $n^{1/d}$ points, and neither of the points is in a subset $S_{i'}^{(j')}$ with $j' < j$.
3. For every subset $S_i^{(j)}$, and for every pair p, q in $S_i^{(j)}$, search in T whether the other $2^d - 2$ vertices of one of the 2^{d-1} cubes defined by p and q are also in S. If so, answer yes, otherwise, answer no.

This leads to the following result.

Theorem 3 *Given a set S of n points in d-dimensional space, it can be decided in time $O(n^{1+1/d} \log n)$ and $O(n)$ space whether S contains a subset of 2^d points that form the vertices of an axis-parallel cube.*

Proof: Follows in the same way as theorem 2. □

3 Axis-parallel rectangles

In this section we first give upper and lower bounds on the maximal number of axis-parallel rectangles which all have their vertices in a set of n points. Then we study the problem of determining whether a planar set of points contains four points that are the vertices of an axis-parallel rectangle. Finally we consider the more-dimensional case.

Theorem 4 *For a set S of n points in d-dimensional space, the maximal number of subsets of 2^d points that form the vertices of an axis-parallel hyperrectangle is $\Theta(n^2)$.*

Proof: To prove the lower bound in the planar case, consider a grid G of size $\lfloor n/2 \rfloor \times 2$. On grid G, one can choose two sides of a rectangle in $\lfloor n/2 \rfloor \cdot (\lfloor n/2 \rfloor - 1)/2 = \Omega(n^2)$ ways. Every pair of sides gives rise to a unique rectangle. The more-dimensional case is similar, using a $\lfloor n/2^{d-1} \rfloor \times 2 \times \cdots \times 2$ grid.

To prove the upper bound in arbitrary dimensional space, observe that any axis-parallel rectangle is determined uniquely by two opposite points that have no coordinates equal. Also, for every rectangle there must be such a pair. The bound follows, since there are $n(n-1)/2$ pairs of points. □

The proof of the above theorem leads to a straightforward $O(n^2 \log n)$ time algorithm for arbitrary dimensional space. Just take any pair of points of which all coordinates are different, and test if the other points are also in the set. We next give a subquadratic time solution for the problem.

Let S be a planar set of n points. Like for squares, the idea is to partition S in a number of subsets of points with equal-valued first coordinate. Notice that S

contains the vertices of an axis-parallel rectangle if and only if there are two different subsets that each contain two points, such that the two points in one subset have second coordinates equal to the second coordinates of the two points in the other subset. We distinguish two types of subsets: the subsets with at most $\sqrt{n}$ points are called *small*, and the subsets with more than $\sqrt{n}$ points are called *large*. Now we can distinguish three cases. Firstly, two small subsets can contain the vertices of a rectangle. Secondly, a small subset and a large subset can each contain two of the vertices of a rectangle, and thirdly, two large subsets can contain the vertices of a rectangle. Again, using lemma 1, we can show that the last case can be avoided.

To solve the first case we need a pointer structure, which is described next. Let $S_1, \ldots, S_k$ be the small subsets, and let S' be their union.

- Let Y be the list containing all different second coordinates of points in S', ordered by increasing value. With each element in Y (value of the second coordinate), an integer *count* is stored, initially zero, and a reference $first$ that points to a list of all points with this second coordinate, sorted by first coordinate.

- Each point p in S' corresponds to a node in the structure with three references, $next_1$, $next_2$ and *back*. $next_1$ leads to the node corresponding to the first point one reaches if one moves in the direction of greater first coordinate (thus a point with equal second coordinate). More formally, the $next_1$ reference of a point $p = (p_1, p_2)$ refers to a point $q = (q_1, q_2)$ if and only if $p_2 = q_2$, $p_1 < q_1$ and for all $r = (r_1, r_2)$, $r_2 = p_2$ implies that $r_1 \leq p_1$ or $r_1 \geq q_1$. Analogously, the reference $next_2$ refers to the point with equal first coordinate and larger second coordinate. The reference *back* of a point $p = (p_1, p_2)$ leads to the element corresponding to p_2 in the list Y. The reference $first$ in the list Y corresponding to some value y of the second coordinate, refers to the point with y as second coordinate, and smallest first coordinate among these points (see figure 1).

Solving the rectangle problem for the small subsets requires building the structure and traversing it in a special way, as given by the algorithm below.

To solve the second case, we count for every small subset S_i and every large subset L_j the number of times a point in S_i has its second coordinate equal to a point in L_j. If this number is more than one, then a rectangle is found. This is done for one small subset and all large subsets simultaneously.

As we have already noticed we will avoid the third case. Let L' be the set of all points in large subsets. Switch the roles of their first and second coordinates and repeat the algorithm. The new subsets will have at most $\sqrt{n}$ points, according to lemma 1. Thus this case is treated as the first case.

Rectangle (S)

1. Partition S into subsets $S_1, \ldots, S_k, L_1, \ldots, L_m$ such that two points are in the same subset if and only if they have equal first coordinate. Furthermore, let

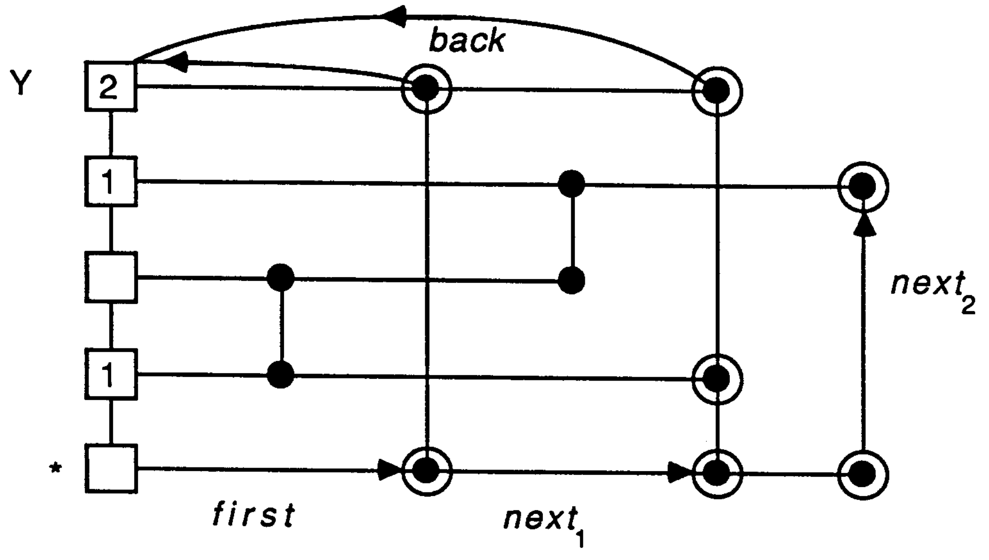

Figure 1: Pointer structure for testing the small subsets.

S_i $(1 \leq i \leq k)$ be the subsets with at most $\sqrt{n}$ points (the small subsets), and let L_j $(1 \leq j \leq m)$ be the subsets with more than $\sqrt{n}$ points (the large subsets).

2. Search for a rectangle in small subsets in the following way. Build the pointer structure as described above.

   The structure is traversed as follows. Traverse the list Y beginning at the element with the smallest value, and for each element (which defines the bottom line of some candidate rectangle), do the following. Follow the reference *first* and then follow the reference $next_1$ of every point p we reach, until we can go no further. For every such point p (which is a point on the bottom line), follow references $next_2$ to points q until we can go no further. For all points q we reach (which are points that lie above a point on the bottom line), follow the *back* reference to an element in the list Y (which defines the top line of a candidate rectangle) and increase *count* with one. If it becomes two, stop and report that S contains the vertices of a rectangle (because two top vertices on the same top line are found, which both must lie above a bottom vertex). (In figure 1, all points that are visited when the algorithm is at the marked element in the list Y, are encircled. Also, the counter values are shown.)

   Before we go to the next element in the list Y, traverse the structure again in the same way and initialize all counters to zero again.

3. Search for a rectangle with two vertices in a small subset and two vertices in a large subset.

   - Let V be the set of all different second coordinates of points in $\bigcup L_j$. Store them ordered in an array A. With each index, corresponding to some value y of the second coordinate, store all names j of large subsets L_j that contain a point with y as second coordinate.

- Make an array B of counters, with one entry for each large subset. Initialize them to zero.
- For every small subset S_i ($1 \leq i \leq k$), compute $S_i \cap L_j$ for all j ($1 \leq j \leq m$) simultaneously in the following way. For each point $p = (p_1, p_2)$ in S_i, search with p_2 in the array A to find all large subsets L_j that have a point with p_2 as second coordinate. For every L_j thus found, increment the counter in the array B at index j. If any counter becomes 2, then stop and report the existence of the vertices of a rectangle in S. Otherwise, reinitialize the counters in array B to zero, before starting on the next small subset.

4. Search for a rectangle with four vertices in two large subsets in the following way. Let L' be the set of all points in large subsets. Switch the roles of first and second coordinates of these points, and use the same method as above to solve the problem. As stated above, only small subsets will appear.

Theorem 5 *Given a set S of n points in the plane, it can be decided in $O(n\sqrt{n})$ time and $O(n)$ space whether S contains a subset of four points that form the vertices of an axis-parallel rectangle.*

Proof: The correctness of the algorithm follows from the above discussion, and the $O(n)$ space bound is straightforward. It remains to analyse the time complexity of the steps taken by the algorithm.

The first step, partitioning the set S of points, can be performed in time $O(n \log n)$ with a straightforward algorithm.

For the second step (the construction and traversal of the pointer structure), we observe that construction can easily be done in $O(n \log n)$ time. All time spent on traversal of the structure can be charged to the number of times every node (point) is visited during the traversal. A point will only be visited once via a $first$ reference or a $next_1$ reference, namely only when the second coordinate of the point is equal to the second coordinate of the element in list Y that is being considered. A point will be visited via a $next_2$ reference once for each point with equal first coordinate and smaller second coordinate. As all points connected via $next_2$ references form one small subset, any point will be visited at most $\sqrt{n}$ times via a $next_2$ reference. Thus all points together are visited $O(n\sqrt{n})$ times. As reinitialization is done in the same way as the traversal, this will at most double the time spent.

For the third step we consider how much time the algorithm takes for each of the three substeps. The first substep can be performed in $O(n \log n)$ with a straightforward algorithm. The second substep takes time $O(m)$, where m is the number of large subsets. For the third substep we consider a small subset S_i. With each point in S_i we search in array A, and possibly increment a number of counters in array B. The searching takes $O(|S_i| \log n)$ time and incrementing counters in B

at most $O(m)$ time, because we can increment only m counters before a rectangle is found and we can terminate. Hence, the total time for all small subsets together is

$$O(\sum_{i=1}^{k}(|S_i|\log n + m)) = O(\sum_{i=1}^{k}(|S_i|\log n + \sqrt{n})) = O(n\sqrt{n}).$$

The time spent on the fourth step follows directly from the analysis of the time spent on the second step (as the new subsets contain at most $\sqrt{n}$ points). □

Before continuing with the more-dimensional case, we show that the problem of deciding whether a planar set of points contains the vertices of an axis-parallel rectangle, is equivalent to the problem of deciding whether a bipartite graph contains a $K_{2,2}$ subgraph. (That is, a complete bipartite graph with $2+2$ nodes and thus four edges. In other words, a cycle of length 4.) The correspondence is as follows. Let S be a planar set of points. Let A be the set of all different first coordinates of the points in S, and let B be the set of all different second coordinates. A and B are the sets of nodes in the bipartite graph. Node $a \in A$ and node $b \in B$ give rise to an edge (a,b) if and only if (a,b) is a point in the set S. The correspondence goes in both directions. With this correspondence, it can easily be seen that S contains the vertices of an axis-parallel rectangle if and only if the bipartite graph consisting of the sets of nodes A, B and edges as defined above contains a $K_{2,2}$ subgraph. With our algorithm, one can obtain an $O(|E|\sqrt{|E|})$ time algorithm for deciding whether a bipartite graph (V,E) (where V is the set of nodes and E is the set of edges) contains a $K_{2,2}$ subgraph.

In [3], Clarkson et al. prove that there exists a c such that any (A,B) bipartite graph with n nodes and more than $c \cdot n\sqrt{n}$ edges, contains a $K_{2,2}$ subgraph. Their result is a variant of a graph theoretic extremal result by Kövári, Sós and Turán [6]. This leads to:

Theorem 6 *Given a bipartite graph $G = (V,E)$, it can be decided in $O(min(|E|^{3/2}, |V|^{9/4}))$ time and $O(|V|+|E|)$ space whether G contains a $K_{2,2}$ subgraph.*

The bound is not new. In [1], Chiba and Nishizeki give an $O(|E|\sqrt{|E|})$ time algorithm for cycles of length four in arbitrary graphs.

Next we consider the more-dimensional axis-parallel rectangle problem: Given a set S of n points in d-dimensional space, does S contain a subset of 2^d points that are the vertices of an axis-parallel hyperrectangle. Contrary to the more-dimensional cube problem, we will not partition S in subsets with only the last coordinate different, but in subsets with only the first coordinate equal. Thus we obtain a partition of S in a number of hyperplanes of which the normal is parallel to the first coordinate axis. Again we distinguish small subsets, which contain at most $n^{1-1/d}$ points, and large subsets, which contain more than $n^{1-1/d}$ points. The d-dimensional problem now reduces to finding hyperfaces (which are $(d-1)$-dimensional hyperrectangles) in the subsets, and then finding two different subsets which both contain the same hyperface, only taking into consideration the first $d-1$ coordinates. Thus we search for opposite $(d-1)$-dimensional hyperrectangles. As before, we first prove a lemma which shows that we need not solve the problem of finding a hyperrectangle with both hyperfaces in large subsets.

Lemma 3 *Given a set S of n (distinct) points in d-dimensional space, then there exists a hyperplane with its normal parallel to one of the coordinate axes and which contains at least one and no more than $n^{1-1/d}$ points of S.*

Proof: Let m_i $(1 \leq i \leq d)$ be the number of different i^{th} coordinates of all points in S. Let $k = \max_{1 \leq i \leq d}\{m_i\} = m_j$, and let $S_1, \ldots, S_k$ be the subsets of S that lie in hyperplanes with constant j^{th} coordinate that contain at least one point of S. By the choice of k, $n \leq k^d$. Suppose that every subset S_i contains more than $n^{1-1/d}$ points. Then $n = \sum_{1 \leq i \leq k} |S_i| > k \cdot n^{1-1/d} \Rightarrow n > k^d$, a contradiction. Thus there is a subset S_i which contains at most $n^{1-1/d}$ points, and the hyperplane containing this subset with normal parallel to the j^{th} coordinate axis satisfies the demands. □

To solve the more-dimensional rectangle problem, partition S in subsets $S_1, \ldots, S_k, L_1, \ldots, L_m$, such that two points are in the same subset if and only if they have equal first coordinate. Let $S_1, \ldots, S_k$ be the subsets with at most $n^{1-1/d}$ points, and $L_1, \ldots, L_m$ are the subsets with more than $n^{1-1/d}$ points.

To determine if there is a hyperrectangle with both hyperfaces in small subsets we need a generalization of the pointer structure used to solve the two-dimensional rectangle problem. Let S' be the union of all points in small subsets, $S' = \bigcup S_i$.

- Let S'' be the set of points of S' restricted to their last $d-1$ coordinates. Let Y be the list containing all different points in S'', ordered lexicographically increasing. With each element in Y, an integer *count* is stored, initially zero, and a reference *first*.

- Each point p in S' corresponds to a node in the structure with $d+1$ references, $next_1, \ldots, next_d$ and *back*, and each node has a counter. $next_1$ leads to the node corresponding to the first point one reaches if one moves in the direction of greater first coordinate (thus a point with the last $d-1$ coordinates equal). More formally, the $next_1$ reference of a point $p = (p_1, \ldots, p_d)$ refers to a point $q = (q_1, \ldots, q_d)$ if and only if $p_2 = q_2, \ldots, p_d = q_d$, $p_1 < q_1$ and for all $r = (r_1, \ldots, r_d)$, $r_2 = p_2, \ldots, r_d = p_d$ implies that $r_1 \leq p_1$ or $r_1 \geq q_1$. Analogously, the references $next_2, \ldots, next_d$ are defined. The reference *back* of a point $p = (p_1, \ldots, p_d)$ leads to the element corresponding to the point $(p_2, \ldots, p_d)$ in the list Y. The reference *first* in the list Y corresponding to some point $(r_2, \ldots, r_d)$, refers to the point with $(r_2, \ldots, r_d)$ as last $d-1$ coordinates, and smallest first coordinate among these points.

We need a few observations before we give the algorithm.

Observation 1 *Let H be an axis-parallel hyperrectangle in d-dimensional space, where p is the vertex of H with all coordinates smallest and q is the vertex of H with all coordinates largest. Then there are exactly $d!$ different ways of moving from p to q along the edges of H, when we are only allowed to move to a vertex with exactly one coordinate larger than the previous vertex.*

Observation 2 *Given the pointer structure for a set S of points in d-dimensional space as described above, a point p in S, and a point q in S for which all coordinates are (strictly) greater than all coordinates of p. Let $\pi = (\pi_1, \ldots, \pi_d)$ be a permutation of the integers $1, \ldots, d$. S contains the vertices of an axis-parallel hyperrectangle with p and q as opposite vertices if and only if one can move from p to q in the pointer structure for every permutation π, by following first arbitrarily many but at least one $next_{\pi_1}$ reference, then at least one $next_{\pi_2}$ reference, ... , and finally at least one $next_{\pi_d}$ reference.*

We will use the latter observation to find $(d-1)$-dimensional hyperrectangles (hyperfaces) in the small subsets when the pointer structure is traversed. Thus we consider all permutations of the coordinates $(2, \ldots, d)$, and we find a hyperface if a point q can be reached in $(d-1)!$ ways from a point p.

To solve the problem of finding a hyperrectangle with one hyperface in a small subset and one in a large subset, we compute the intersections of every small subset with every large subset, when the points are restricted to their last $d-1$ coordinates. Then we repeat the algorithm recursively in $(d-1)$-dimensional space.

Hyperrectangle (S, d)

1. Let $N = n^{1-1/d}$.

2. Partition S into subsets $S_1, \ldots, S_k, L_1, \ldots, L_m$ such that two points are in the same subset if and only if they have equal first coordinate. Furthermore, let S_i $(1 \leq i \leq k)$ be the subsets with at most N points (the small subsets), and let L_j $(1 \leq j \leq m)$ be the subsets with more than N points (the large subsets).

3. Search for a hyperrectangle with both hyperfaces in small subsets in the following way. Build the pointer structure as described above.

   The structure is traversed as follows. Traverse the list Y in order, and for each element, do the following. Follow the reference $first$ and then follow the reference $next_1$ of every point p we reach, until we can go no further. For every such point p (which lies in a small subset, where we will search for hyperfaces) and for every permutation $\pi = (\pi_2, \ldots, \pi_d)$ of the numbers $(2, \ldots, d)$, follow every possible number but at least one $next_{\pi_2}$ reference, then at least one $next_{\pi_3}$ reference, ..., and finally at least one $next_{\pi_d}$ reference. For every point q we reach, given a permutation, increment the counter of that point. If it becomes $(d-1)!$, there are $(d-1)!$ ways of reaching it from p and, according to observation 2, p and q must form the opposite vertices of a $(d-1)$-dimensional hyperface. Hence, we follow the *back* reference of q and increment the corresponding counter in the list Y. If this counter in Y becomes two, two matching hyperfaces are found and we can stop and report that S contains the vertices of a hyperrectangle.

Before we go to the next element in the list Y, traverse the structure again in the same way and reinitialize all counters to zero.

4. Search for a hyperrectangle with one hyperface in a small subset and one hyperface in a large subset in the following way.

   - Let V be the set of all different points in $\bigcup L_j$, restricted to their last $d-1$ coordinates. Store them lexicographically ordered in an array A. With each index corresponding to some point $p = (p_2, \ldots, p_d)$, store all names j of large subsets L_j that contain a point with $(p_2, \ldots, p_d)$ as last $d-1$ coordinates.
   - Make an array B of lists, with one entry for each large subset. Initialize each entry to the empty list.
   - For every small subset S_i $(1 \le i \le k)$, compute $S_i \cap L_j$ for all j $(1 \le j \le m)$ simultaneously in the following way. For each point $p = (p_1, \ldots, p_d)$ in S_i, search with $(p_2, \ldots, p_d)$ in the array A to find all large subsets L_j that have a point with $(p_2, \ldots, p_d)$ as last $d-1$ coordinates. For every L_j thus found, store in the array B at index j the point $(p_2, \ldots, p_d)$. When all points in S_i are treated this way, then solve the m $(d-1)$-dimensional problems of finding a $(d-1)$-dimensional hyperrectangle for each set B_j, which are the points in the intersections of the large subset L_j with S_i. This is done recursively. If a $(d-1)$-dimensional hyperrectangle is found, it exists in both S_i and L_j and, hence, a hyperrectangle is found.

     Reinitialize all lists in the array B before continuing with the next small subset.

5. Search for a hyperrectangle with both hyperfaces in two large subsets in the following way. Let L' be the set of all points in large subsets. Switch the roles of the coordinates (that is, let another coordinate take the role of the first coordinate) and repeat the algorithm at step 2. (From lemma 3, there will only be small subsets left after $d-2$ times of switching the roles of coordinates.)

Theorem 7 *Given a set S of n points in E^d, it can be decided in $O(n^{2-1/d})$ time and $O(n)$ space whether S contains a subset of 2^d points that form the vertices of an axis-parallel hyperrectangle.*

Proof: It is not difficult to see that the algorithm uses $O(n)$ space. For the time bound, we prove by induction on d that the solution given above takes $O(n^{2-1/d})$ time. Let $N = n^{1-1/d}$.

If $d = 2$, then the time bound follows from theorem 5.

If $d > 2$, then distinguish the steps of the algorithm above.

The first two steps can easily be done in $O(n \log n)$ time.

For the third step, notice that every small set S_i is visited at most $|S_i|$ times, and at each visit, any point in S_i is visited at most a constant number of times (dependent on d, namely $(d-1)!$). Thus traversing the structure takes time bounded by

$$c \cdot \sum_{1 \leq i \leq k} |S_i|^2 \leq c \cdot \sum_{1 \leq i \leq k} N \cdot |S_i| \leq c \cdot N \cdot n = O(n^{2-1/d})$$

(where c is some constant).

The first and second substeps of step 4 clearly take $O(n \log n)$ time. Computing the intersection of one small subset S_i with all large subsets $L_1, \ldots, L_m$ takes time bounded by $c \cdot |S_i| \cdot (m + \log n)$, thus for all small subsets we spend time bounded by

$$c \cdot \sum_{1 \leq i \leq k} |S_i| \cdot (m + \log n) \leq c \cdot \sum_{1 \leq i \leq k} |S_i| \cdot (n/N + \log n) \leq$$
$$c \cdot (n^2/N + n \log n) = O(n^{1+1/d}).$$

Solving the $(d-1)$-dimensional problem for $S_i \cap L_j$ takes time bounded by $c \cdot |S_i \cap L_j|^{2-1/(d-1)}$ by induction. Thus the total time spent to solve all $(d-1)$-dimensional problems is bounded by

$$c \cdot \sum_{1 \leq i \leq k} \sum_{1 \leq j \leq m} |S_i \cap L_j|^{2-1/(d-1)} \leq c \cdot \sum_{1 \leq i \leq k} m \cdot |S_i|^{2-1/(d-1)} \leq$$
$$c \cdot n/N \cdot \sum_{1 \leq i \leq k} |S_i|^{2-1/(d-1)} \leq c \cdot n \cdot N^{-1/(d-1)} \cdot \sum_{1 \leq i \leq k} |S_i| \leq$$
$$c \cdot n^2 \cdot N^{-1/(d-1)} = O(n^{2-1/d}).$$

The fourth step (and hence also the other steps) can only occur d times by lemma 3, and this step takes $O(n)$ time. □

Observe that the problem of deciding whether a d-dimensional set S of points contains the vertices of an axis-parallel hyperrectangle, is equivalent to the graph problem of deciding whether a d-partite graph contains a $K_{2,\ldots,2}$ subgraph (d two's).

4 Arbitrarily oriented squares and rectangles

In this section we study the problems dealt with in the previous sections, but this time for not necesarily axis-parallel squares and rectangles. Again we begin with giving upper and lower bounds on the maximal number of squares and rectangles that can occur in a planar set of points. Then we give algorithms for recognizing (hyper-)squares and (hyper-)rectangles in sets of points in d-dimensional space. Our algorithms take time $O(n^d \log n)$ for rectangles, and time $O(n^2 \log n)$, $O(n^3)$ and $O(n^{d-1/2}\beta(n))$ for $d = 2$, $d = 3$ and $d \geq 4$, respectively, for squares. ($\beta(n)$ is related to the inverse of Ackermann's function.)

Theorem 8 *For a set S of n points in the plane, the maximal number of subsets of four points that form the vertices of a square is $\Theta(n^2)$.*

Proof: To prove the lower bound, place the points of S on a grid G of size $\lfloor\sqrt{n}\rfloor \times \lfloor\sqrt{n}\rfloor$. Divide the grid in sixteen equal-sized square subgrids. Let p be a point in the section just above the bottom left section, and let q be a point in the section just to the right of the bottom left section. Then p and q determine a unique square of which all vertices are on the grid. As there are $\Omega(n)$ places to put p and $\Omega(n)$ places to put q, there are $\Omega(n^2)$ squares on the grid.

The upper bound is trivial because any two points considered as opposite vertices uniquely determine a square. $\square$

Theorem 9 *For a set S of n points in the plane, the maximal number of subsets of four points that form the vertices of a rectangle is $\Omega(n^2)$ and $O(n^2\sqrt{n})$.*

Proof: The lower bound follows immediately from previous results for the axis-parallel case. The upper bound is proved next.

For two points p and q, let $C_{p,q}$ be the circle containing p and q with diameter $|\overline{pq}|$. It is easy to see that four points p_1, p_2, p_3, p_4 are the vertices of a rectangle (with diagonals $\overline{p_1p_3}$ and $\overline{p_2p_4}$) if and only if $C_{p_1,p_3} = C_{p_2,p_4}$.

Let $p_1, \ldots, p_n$ be the points of S. Let them define the set of different circles $\{C_1, \ldots, C_m\}$ and let d_i be the number of opposite pairs of points on C_i, i.e., $d_i = |\{(p_j, p_k) : 1 \leq j < k \leq n \text{ and } C_i = C_{p_j,p_k}\}|$. The number of rectangles of S is given by $\sum_{i=1}^{m} d_i(d_i - 1)/2$. Furthermore, we have $\sum_{i=1}^{m} d_i = n(n-1)/2$, because every pair (p_j, p_k) contributes to only one d_i, and $d_i \leq n/2$ $(1 \leq i \leq m)$.

Assume without loss of generality that $d_1 \geq d_2 \geq \cdots \geq d_m$, and let k be such that $d_k > 2\sqrt{n}$, $d_{k+1} \leq 2\sqrt{n}$. As two circles intersect in at most two points, we have $k \leq \sqrt{n}$. (If $k > \sqrt{n}$, then the number of points in S would be at least $\sum_{i=1}^{k}(2d_i - 2(i-1)) > 4k\sqrt{n} - k(k-1) > 4n - (n + \sqrt{n}) > n$, a contradiction.) Now we have

$$\sum_{i=1}^{m} d_i(d_i - 1)/2 \leq \sum_{i=1}^{m} d_i^2 = \sum_{i=1}^{k} d_i^2 + \sum_{i=k+1}^{m} d_i^2 \leq$$

$$\sum_{i=1}^{\sqrt{n}} d_i^2 + \sum_{i=k+1}^{m} (2\sqrt{n})d_i \leq \sum_{i=1}^{\sqrt{n}} (n/2)^2 + 2\sqrt{n} \cdot n(n-1)/2 \leq 2n^2\sqrt{n}.$$

$\square$

Next we address the problem of finding the vertices of hyperrectangles and hypersquares in a set of points in d-dimensional space. For brevity we will speak of rectangles and squares in d-space. First some simple properties and observations on rectangles, squares and sets of points in d-space are given.

Observation 3 *A rectangle or square has 2^d vertices.*

Observation 4 *A rectangle or square has 2^{d-1} (parallel) edges for which the bisecting hyperplanes coincide. In this case, the 2^d endpoints of the edges form all vertices of the rectangle or square.*

Observation 5 *If there are 2^{d-1} pairs of points for which the bisecting hyperplanes coincide, and the distances between the two points in a pair are equal for every pair, then the 2^d points are the vertices of a rectangle in d-space if and only if the projections of the points on the bisecting hyperplane h are the vertices of a rectangle on h ($(d-1)$-space). The 2^d points are the vertices of a square in d-space if and only if the projections of the points on the bisecting hyperplane h are the vertices of a square on h with edge length equal to the distances between the two points in the pairs.*

The above observations suggest the following algorithms, which both take every pair of points and bucket (classify) the pairs, giving rise to a simpler subproblem for every bucket.

Rectangle (S, d)

1. If $d = 1$, then report the existence of a rectangle if $|S| > 1$ and terminate.
2. If $d > 1$, then bucket every pair of points of S, such that two pairs (p_1, p_2) and (q_1, q_2) are in the same bucket if and only if the bisecting hyperplane of p_1 and p_2 is equal to the bisecting hyperplane of q_1 and q_2, and the distance between p_1 and p_2 is equal to the distance between q_1 and q_2. Let the buckets be $B_1, \ldots, B_m$.
3. For every bucket B_i corresponding to some hyperplane h, project one point of every pair in B_i on h, and solve the $(d-1)$-dimensional problem of finding a rectangle in the projected points on h recursively.

Square (S, d, e) (e is the edge length, which is not specified initially)

1. If $d = 1$ and the edge length is not specified, then report the existence of a square if $|S| > 1$ and terminate. If $d = 1$ and the edge length is specified, then report the existence of a square if there is a pair of points in S with the specified edge length, and terminate.
2. If $d > 1$ and the edge length is specified, then only consider pairs of points with distance e.

   Bucket every pair of points of S, such that two pairs (p_1, p_2) and (q_1, q_2) are in the same bucket if and only if the bisecting hyperplane of p_1 and p_2 is equal to the bisecting hyperplane of q_1 and q_2, and the distance between p_1 and p_2 is equal to the distance between q_1 and q_2. Let the buckets be $B_1, \ldots, B_m$.

3. For every bucket B_i corresponding to some hyperplane h and some distance x, project one point of every pair in B_i on h, and solve the $(d-1)$-dimensional problem of finding a square with edge length x in the projected points on h recursively.

Theorem 10 *Given a set S of n points in d-dimensional space ($d \geq 2$), it can be decided in $O(n^d \log n)$ time and $O(n^2)$ space whether S contains a subset of 2^d points that form the vertices of a rectangle. It can be decided in $O(n^2 \log n)$ time ($d = 2$), $O(n^3)$ time ($d = 3$), or $O(n^{d-1/2}\beta(n))$ time ($d \geq 4$), and $O(n^2)$ space whether S contains a subset of 2^d points that form the vertices of a square ($\beta(n)$ is related to the extremely slowly growing inverse of Ackermann's function).*

Proof: Let $T(d,n)$ denote the time taken by the above algorithm for rectangles in a set of n points in d-space, and let $|B_i|$ be the number of pairs of points in bucket B_i. Then

$$T(d,n) \leq c \cdot n^2 \log n + \sum_{i=1}^{m} T(d-1, |B_i|),$$
$$T(1,n) = O(n).$$

Using the fact that $|B_i| \leq n/2$ and $\sum_{i=1}^{m} |B_i| \leq n(n-1)/2$, one can prove that $T(d,n) = O(n^d \log n)$ for $d \geq 2$.

Next we consider the algorithm for squares. Let $T(d,n)$ denote the time taken by the square algorithm when the edge length is not specified, and $T^*(d,n)$ the time taken when the edge length is specified. Furthermore, let $M(d,n)$ be the maximal number of pairs of points in a set of n points in d-space which lie some specified distance from each other. Then

$$T(d,n) \leq c \cdot n^2 \log n + \sum_{i=1}^{m} T^*(d-1, |B_i|),$$

$$T^*(d,n) \leq c \cdot n^2 + c \cdot M(d,n) \log n + \sum_{i=1}^{m} T^*(d-1, |B_i|),$$
$$T^*(1,n) = O(n \log n),$$

and we have $|B_i| \leq n/2$, $\sum |B_i| \leq n(n-1)/2$ when the edge length is not specified, and $\sum |B_i| \leq M(d,n)$ when the edge length is specified.

The problem of determining $M(d,n)$ is called the unit-distance problem and was posed by Erdös [4, 5]. The best known bounds are $M(2,n) = O(n^{4/3})$ and $M(3,n) = O(n^{3/2}\beta(n))$ ($\beta(n)$ as in the theorem), see [3, 7]. It is known that $M(d,n) = \Theta(n^2)$ for $d \geq 4$, see [2]. Now we can prove $T^*(2,n) = O(n^2)$ and $T^*(d,n) = O(n^{d-1/2}\beta(n))$ for $d \geq 3$. The bounds of the theorem follow. □