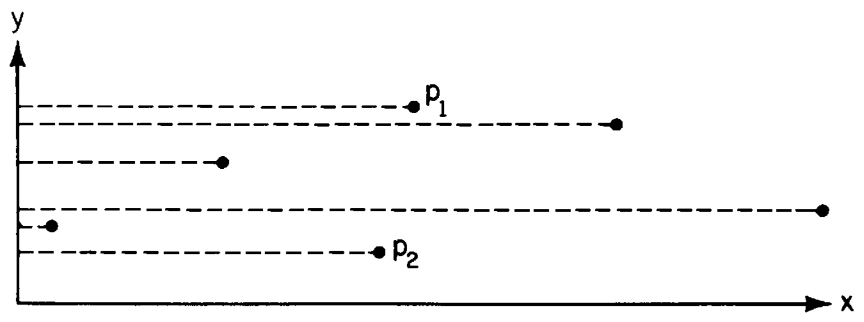

Figure 5.8 The failure of projection methods. Points p_1 and p_2 are the closest pair but are farthest in y-distance.

5.4 The Closest Pair Problem: A Divide-and-Conquer Approach

The lower bound of Theorem 5.2 challenges us to find a $\theta(N \log N)$ algorithm for CLOSEST PAIR. There seem to be two reasonable ways to attempt to achieve such behavior: a direct recourse to sorting or the application of the divide-and-conquer scheme. We can readily dispose of the former, since the environment where sorting is useful is a total ordering. The only reasonable way to obtain a total ordering seems to be to project all the points on a straight line: unfortunately projection destroys essential information, as illustrated informally in Figure 5.8: points p_1 and p_2 are the closest, but they are farthest when projected on the y-axis.

A second way toward a $\theta(N \log N)$ performance is to split the problem into two subproblems whose solutions can be combined in linear time to give a solution to the entire problem [Bentley–Shamos (1976); Bentley (1980)]. In this case, the obvious way of applying divide-and-conquer does not lead to any improvement, and it is instructive to explore why it fails. We would like to split the sets into two subsets, S_1 and S_2, each having about $N/2$ points, and obtain a closest pair in each set recursively. The problem is how to make use of the information so obtained. The possibility still exists, though, that the closest pair in the set consists of one element of S_1 and one element of S_2, and there is no clear way to avoid making $N^2/4$ additional comparisons. Letting $P(N, 2)$ denote the running time of the algorithm to find the closest pair in two dimensions, the preceding observations lead to a recurrence of the form

$$P(N, 2) = 2P(N/2, 2) + O(N^2)$$

whose solution is $P(N, 2) = O(N^2)$. Let us try to remedy the difficulty by retreating to one dimension.

The only $\theta(N \log N)$ algorithm we know on the line is the one which sorts the points and performs a linear-time scan. Since, as noted above, sorting will not generalize to two dimensions, let us try to develop a one-dimensional

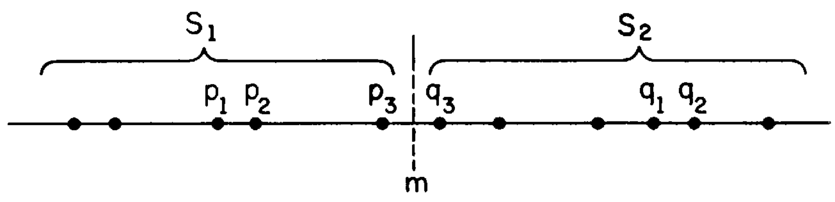

Figure 5.9 Divide-and-conquer in one dimension.

divide-and-conquer scheme that will. Suppose we partition a set of points on the line by some point m, into two sets S_1 and S_2 with the property that $p < q$ for all $p \in S_1$ and $q \in S_2$. Solving the closest pair problem recursively on S_1 and S_2 separately gives us two pairs of points $\{p_1, p_2\}$ and $\{q_1, q_2\}$, the closest pairs in S_1 and S_2, respectively. Let δ be the smallest separation found thus far (see Figure 5.9):

$$\delta = \min(|p_2 - p_1|, |q_2 - q_1|).$$

The closest pair in the whole set is either $\{p_1, p_2\}$, $\{q_1, q_2\}$ or some $\{p_3, q_3\}$, where $p_3 \in S_1$ and $q_3 \in S_2$. Notice, though, *and this is the key observation,* that both p_3 and q_3 must be within distance δ of m if $\{p_3, q_3\}$ is to have a separation smaller than δ. (It is clear that p_3 must be the rightmost point in S_1 and q_3 is the leftmost point in S_2, but this notion is not meaningful in higher dimensions so we wish to be somewhat more general.) How many points of S_1 can lie in the interval $(m - \delta, m]$? Since every semi-closed interval of length δ contains at most one point of S_1, $(m - \delta, m]$ contains at most one point. Similarly, $[m, m + \delta)$ contains at most one point. The number of pairwise comparisons that must be made between points in different subsets is thus at most *one*. We can certainly find all points in the intervals $(m - \delta, m]$ and $[m, m + \delta)$ in linear time, so an $O(N \log N)$ algorithm results (CPAIR1).

```
    function CPAIR1($S$)
    Input: $X[1:N]$, $N$ points of $S$ in one dimension.
    Output: $\delta$, the distance between the two closest.
begin if ($|S| = 2$) then $\delta := |X[2] - X[1]|$
    else if ($|S| = 1$) then $\delta := \infty$
        else begin $m :=$ median($S$);
            Construct($S_1, S_2$) (*$S_1 = \{p: p \le m\}, S_2 = \{p: p > m\}$*);
            $\delta_1 :=$ CPAIR1($S_1$);
            $\delta_2 :=$ CPAIR1($S_2$);
            $p := \max(S_1)$;
            $q := \min(S_2)$;
            $\delta := \min(\delta_1, \delta_2, q - p)$
        end;
    return $\delta$
end.
```

This algorithm, while apparently more complicated than the simple sort and scan, provides the necessary transition to two dimensions.

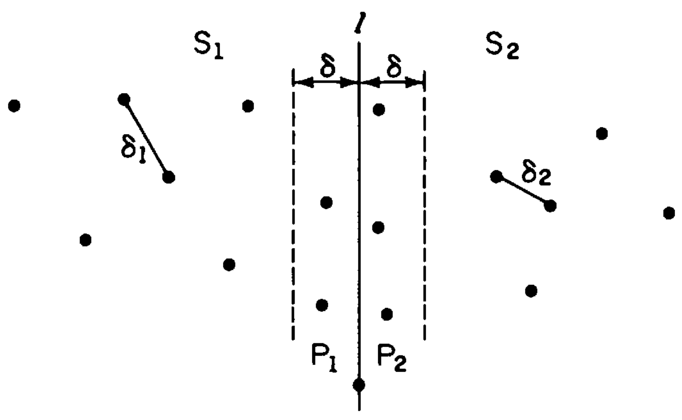

Figure 5.10 Divide-and-conquer in the plane.

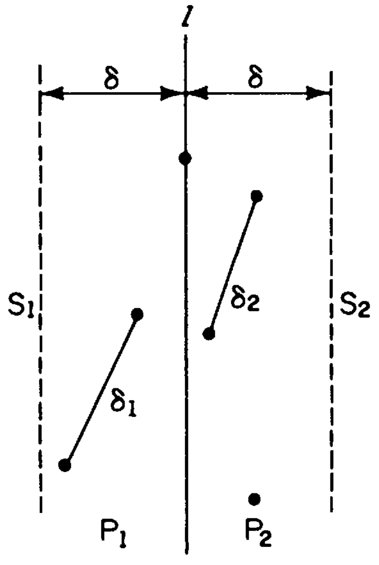

Figure 5.11 All points may lie within δ of l.

Generalizing as directly as possible, let us partition a two-dimensional set S into two subsets S_1 and S_2, such that every point of S_1 lies to the left of every point of S_2. That is, we cut the set by a vertical line l defined by the median x-coordinate of S. Solving the problem on S_1 and S_2 recursively, we obtain δ_1 and δ_2, the minimum separations in S_1 and S_2, respectively. Now let $\delta = \min(\delta_1, \delta_2)$. (See Figure 5.10.)

If the closest pair consists of some $p \in S_1$ and some $q \in S_2$, then surely p and q are both within distance δ of l. Thus, if we let P_1 and P_2 denote two vertical strips of width δ to the left and to the right of l, respectively, then $p \in P_1$ and $q \in P_2$. At this point complications arise that were not present in the one-dimensional case. On the line we found at most one candidate for p [8] and at most one for q. In two dimensions, every point can be a candidate because it is only necessary for a point to lie within distance δ of l. Figure 5.11 shows a set with this property. It again seems that $N^2/4$ distance comparisons will be

[8] In CPAIR1 there is *exactly* one candidate for p: $p = \max(S_1)$.

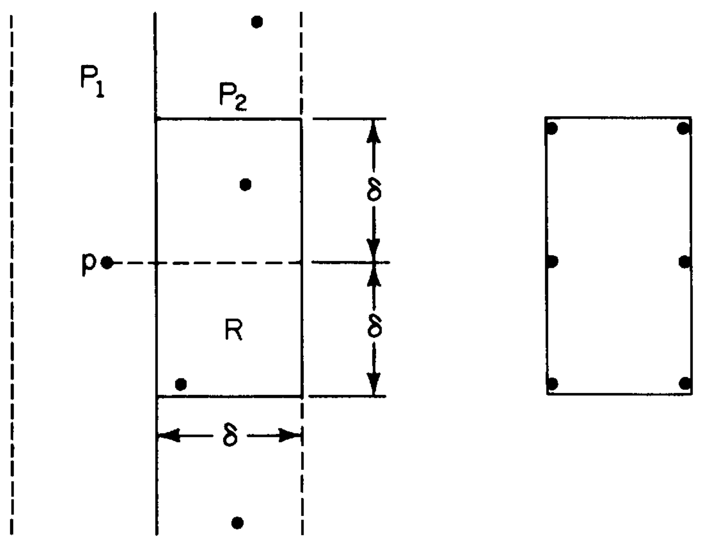

Figure 5.12 For every point in P_1 only a constant number of points in P_2 need to be examined (at most six).

required to find the closest pair, but we will now show that the points lying within the strips of width δ around l have special structure indeed.

Referring now to Figure 5.12, consider any point p in P_1. We must find all points q in P_2 that are within δ of p, but how many of these can there be? They must lie in the $\delta \times 2\delta$ rectangle R and we know that no two points in P_2 are closer together than δ.[9] The maximum number of points of separation at least δ that can be packed into such a rectangle is six, as shown in the figure. This means that for each point of P_1 we need only examine at most six points of P_2, not $N/2$ points. In other words, at most $6 \times N/2 = 3N$ distance comparisons will need to be made in the subproblem merge step instead of $N^2/4$.

We do not yet have an $O(N \log N)$ algorithm, however, because even though we know that only six points of P_2 have to be examined for every point of P_1, we do not know which six they are! To answer this question, suppose we project p and all the points of P_2 onto l. To find the points of P_2 that lie in rectangle R, we may restrict ourselves to consider the projection points within distance δ from the projection of p (at most six). If the points are sorted by y-coordinate, for *all* points in P_1 their nearest-neighbor "candidates" in P_2 can be found in a single pass through the sorted list. Here is a sketch of the algorithm as developed so far.

```
procedure CPAIR2($S$)
1. Partition $S$ into two subsets, $S_1$ and $S_2$, about the vertical median line $l$.
2. Find the closest pair separations $\delta_1$ and $\delta_2$ recursively.
3. $\delta := \min(\delta_1, \delta_2)$;
4. Let $P_1$ be the set of points of $S_1$ that are within $\delta$ of the dividing line $l$ and let
   $P_2$ be the corresponding subset of $S_2$. Project $P_1$ and $P_2$ onto $l$ and sort by
   $y$-coordinate, and let $P_1^*$ and $P_2^*$ be the two sorted sequences, respectively.
```

[9] This crucial observation is due to H. R. Strong (personal communication, 1974).

5. The "merge" may be carried out by scanning P_1^* and for each point in P_1^* by inspecting the points of P_2^* within distance δ. While a pointer *advances* on P_1^*, the pointer of P_2^* may oscillate within an interval of width 2δ. Let δ_l be the smallest distance of any pair thus examined.
6. $\delta_S := \min(\delta, \delta_l)$.

If $T(N)$ denotes the running time of the algorithm on a set of N points, Steps 1 and 5 take $O(N)$ time, Steps 3 and 6 take constant time, and Step 2 takes $2T(N/2)$ time. Step 4 would take $O(N \log N)$ time, if sorting were to be performed at each execution of the step; however, we can resort to a standard device—called *presorting*—whereby we create once and for all a list of points sorted by y-coordinate, and, when executing Step 4, extract the points from the list *in sorted order* in only $O(N)$ time.[10] This trick enables us to write the recurrence for the running time $P(N, 2)$ of the closest-pair algorithm in two dimensions

$$P(N,2) = 2P(N/2,2) + O(N) = O(N \log N) \tag{5.1}$$

which gives the following theorem.

Theorem 5.3. *The shortest distance determined by N points in the plane can be found in $\theta(N \log N)$ time, and this is optimal.*

The central features of the described divide-and-conquer strategy, which enable its extension to higher-dimensional cases, are summarized as follows:

1. The step at which the subproblem solutions are combined takes place in one lower dimension (from the plane to the line).
2. The two point sets involved in the combination of the subproblem solutions have the property of *sparsity*, i.e., it is guaranteed that in each set the distance between any two points is lower-bounded by a known constant.

The emergence of sparsity in the combination step is really the crucial factor. More formally we say

Definition 5.1. Given real $\delta > 0$ and integer $c \geq 1$, a point set S in d-dimensional space has *sparsity c for given δ*, if in any hypercube of side 2δ [11] there are at most c points of S.

Here sparsity is defined as a (monotone nondecreasing) function of δ. Sparsity can be induced, as we shall see, by choosing δ as the minimum

[10] This algorithm was implemented nonrecursively by Hoey and Shamos and a curious phenomenon was observed: the number of distance computations made was always strictly less than N. That this is always the case will be shown later in this section. Of course, the behavior of the algorithm is still dominated by the sorting step.

[11] Such cube is also called a *box*.

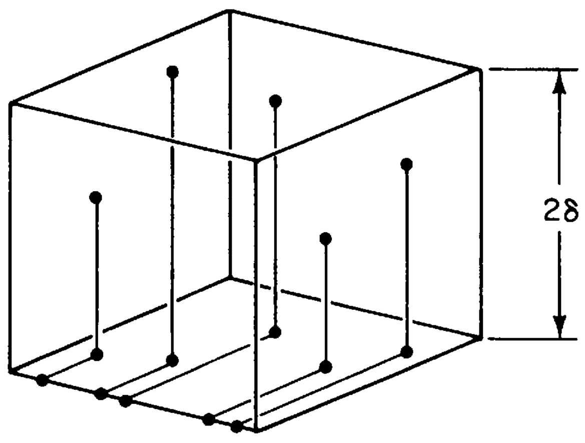

Figure 5.13 Sparsity is preserved through orthogonal projection (for $d \geq 2$).

distance between pairs of points. In the preceding example, the set of points within δ of the dividing line l is sparse, with sparsity $c = 12$, as can be inferred from Figure 5.12. (Note that $c = 12 = 4 \times 3$ for $d = 2$.)

For $d \geq 2$, the portion of E^d contained between two hyperplanes orthogonal to one of the coordinate axes and at distance 2δ from each other is called a *δ-slice*. Note that if a δ-slice has sparsity c for given δ, this sparsity is preserved through projection of the δ-slice on the hyperplane bisecting the δ-slice (a $(d-1)$-dimensional variety). Indeed the d-dimensional cube of side 2δ projects to a $(d-1)$-dimensional cube, also of side 2δ, containing exactly as many points as the original cube (Figure 5.13).

Suppose now to have a set S of N points in E^d. According to the previous pattern, we propose the following algorithm:

```
procedure CPAIR d(S)
1. Partition set $S$ (in time $O(N)$) by means of a suitable hyperplane $l$ (called
   cut-plane) orthogonal to a coordinate axis, into two subsets $S_1$ and $S_2$ of
   sizes $\alpha N$ and $(1-\alpha)N$ respectively (where $0 < \alpha_0 \leq \alpha \leq 1 - \alpha_0$, to be later
   determined).
2. Apply recursively the algorithm to $S_1$ and $S_2$ and obtain the minimum
   distances $\delta_1$ and $\delta_2$ in $S_1$ and $S_2$, respectively, and let $\delta = \min(\delta_1, \delta_2)$.
3. Combine the results by processing a $\delta$-slice of $E^d$ of width $2\delta$ bisected by $l$.
```

We now concentrate on Step 3. Assume, as a working hypothesis, that the set of points in the δ-slice, called S_{12}, has sparsity c for the given δ (we shall later satisfy ourselves that such c exists). Let S_{12}^* be the projection of S_{12} onto the hyperplane l (a $(d-1)$-dimensional space). Clearly, a necessary condition for a pair of points of S_{12} to be the closest pair in S is that their projections in l have distance less than δ. Thus we have a sparse set, S_{12}^*, in which we must find all pairs of points at bounded distance ($<\delta$). This is an instance of the following problem, which is of interest in its own right.

PROBLEM P.7 (FIXED RADIUS NEAREST-NEIGHBOR REPORT IN SPARSE SET). Given real δ and a set of M points in E^d, with sparsity c for given δ, report all pairs at distance less than δ.

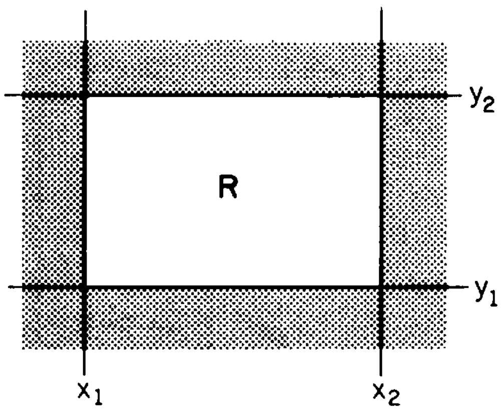

Figure 5.14 Definition of rectangle $R = [x_1, x_2] \times [y_1, y_2]$.

If we let $|S_{12}^*| = M$ and $F_{\delta,c}(M, d)$ denote the running time of Problem P.7, the preceding analysis leads to the recurrence

$$P(N, d) = P(\alpha N, d) + P((1 - \alpha)N, d) + O(N) + F_{\delta,c}(M, d - 1). \quad (5.2)$$

Our objective is the minimization of $F_{\delta,c}(M, d - 1)$. This, in turn, corresponds both to choosing a cut-plane minimizing M, and to developing an efficient algorithm for Problem P.7. Both of these tasks are realizable by virtue of the following interesting theorem.

Theorem 5.4 [Bentley–Shamos (1976)]. *Given a set S of N points in E^d, there exists a hyperplane l perpendicular to one of the coordinate axes with the following properties:*

(i) *both subsets S_1 and S_2 of S on either side of l contain at least $N/4d$ points;*
(ii) *there are at most $dbN^{1-1/d}$ points in a δ-slice of E^d around l, where $\delta = \min(\delta_1, \delta_2)$ and δ_i is the minimum interpoint distance in S_i $(i = 1, 2)$, and b is an upper bound to the number of points with least distance δ contained in a box of side 2δ.*

Proof. We shall carry out the proof for $d = 2$, because of its more immediate intuitive appeal. Its d-dimensional generalization involves the same steps. We assume, with a negligible loss of generality, that N is a multiple of 8. On the x-axis we determine an interval $[x_1, x_2]$, such that there are $N/8$ points of S both to the left of x_1 and to the right of x_2; letting $C_x \subseteq S$ denote the set of points whose abscissae are between x_1 and x_2, we note that $|C_x| = 3N/4$. Similarly, we determine $[y_1, y_2]$ on the y-axis. The Cartesian product $[x_1, x_2] \times [y_1, y_2]$ is called R (Figure 5.14).

On the x-axis we determine the *largest* interval, totally contained in $[x_1, x_2]$ and containing the projections of $2bN^{1/2}$ points; we do likewise on the y-axis. Let γ be the maximum of the lengths of these two intervals. Without loss of generality, we assume that $[x_1', x_2']$ is the subinterval (of $[x_1, x_2]$) yielding γ. We claim that the line l of equation $x = (x_1' + x_2')/2$ (perpendicular bisector of the segment $\overline{x_1'x_2'}$) satisfies properties (i) and (ii) (see Figure 5.15).

To prove this claim, we begin by showing by contradiction that $\gamma < 2\delta$ is

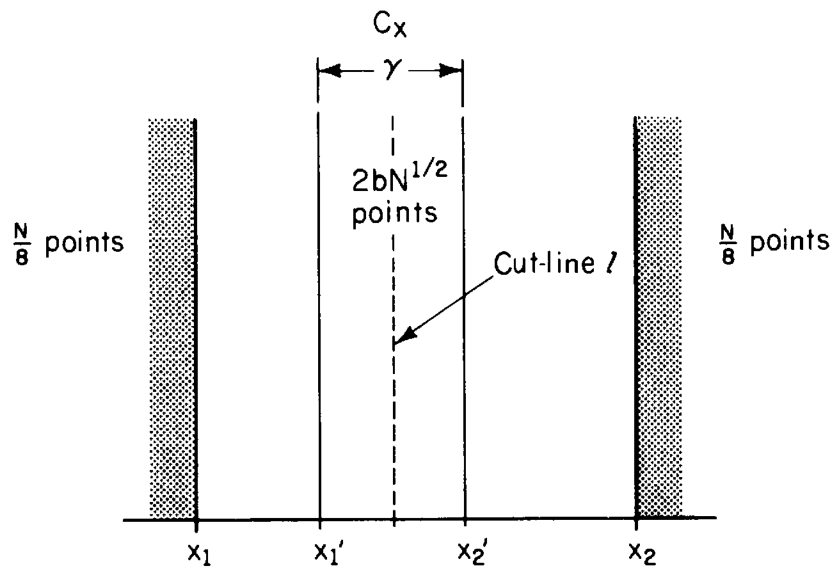

Figure 5.15 Relationship between $[x_1, x_2]$ and $[x_2', x_2']$.

impossible; we shall then see that the condition $\gamma \geq 2\delta$ ensures the desired result.

The condition $\gamma < 2\delta$ means that any horizontal and vertical strip of width 2δ projecting in the interior of $[y_1, y_2]$ and $[x_1, x_2]$, respectively, contains more than $2bN^{1/2}$ points.

If we now partition $[x_1, x_2]$ into subintervals of size 2δ, in each of the vertical strips of the plane projecting to these subintervals there are more than $2bN^{1/2}$ points. Since there are $\lfloor (x_2 - x_1)/2\delta \rfloor$ such subintervals, we have $\lfloor (x_2 - x_1)/2\delta \rfloor \cdot 2bN^{1/2} < |C_x| = 3N/4$, or $\lfloor (x_2 - x_1)/2\delta \rfloor < 3N^{1/2}/8b$. Arguing in the same way for the $[y_1, y_2]$ interval we obtain $\lfloor (y_2 - y_1)/2\delta \rfloor < 3N^{1/2}/8b$. Consider now the rectangle $R = [x_1, x_2] \times [y_1, y_2]$. This rectangle contains at most $\lceil (x_2 - x_1)/2\delta \rceil \cdot \lceil (y_2 - y_1)/2\delta \rceil$ squares of side 2δ. Each such square is exactly a box of side 2δ and so, by the hypothesis of the theorem, it contains at most b points. It follows that the number of points of S in R is at most

$$\left\lceil \frac{x_2 - x_1}{2\delta} \right\rceil \cdot \left\lceil \frac{y_2 - y_1}{2\delta} \right\rceil \cdot b \leq \left(\left\lfloor \frac{x_2 - x_1}{2\delta} \right\rfloor + 1 \right) \left(\left\lfloor \frac{y_2 - y_1}{2\delta} \right\rfloor + 1 \right) \cdot b$$
$$< \left(\frac{3N^{1/2}}{8b} + 1 \right)^2 \cdot b.$$

Since b is a small constant ≥ 1, the right side is maximized by $b = 1$, so that R contains at most $2(3/8)^2 N \cong 0.28N$ points for $N \geq 8$.

On the other hand, the vertical strips external to $[x_1, x_2]$ contain $2 \cdot N/8$ points, and so do the horizontal strips external to $[y_1, y_2]$ (see Figure 5.15). By the principle of inclusion–exclusion, the number of points external to R is at most $4N/8 = N/2$, so that R contains *no less than* $N/2$ points. A contradiction has been obtained whence the condition $\gamma < 2\delta$ is impossible.

Since $\gamma \geq 2\delta$, the result follows, for property (i) is ensured by the construction of $[x_1, x_2]$, and property (ii) holds since the δ-slice around l contains at most $2bN^{1/2}$ points. Therefore line l satisfies the specifications of the theorem. □

We use this theorem to show first that a cut-plane l can be chosen in Step 1 of CPAIR$d(S)$ so that $1/4d \leq \alpha \leq 1 - 1/4d$, and S_{12}, the set of points in the δ-slice around l, has cardinality at most $cdN^{1-1/d}$, where $c = 4 \cdot 3^{d-1}$. Indeed, if $\delta = \min(\delta_1, \delta_2)$, a box of side 2δ in the δ-slice contains at most $c = 4 \cdot 3^{d-1}$ points, as can be easily seen by induction on d. The induction starts with $d = 2$, for which we have already seen that at most 12 points are contained in a "box" of side 2δ. This proves that cut-plane l exists. To determine it, we presort S, once and for all, on *all* coordinates in time $O(dN \log N)$. By a simple $O(dN)$ scan of each of the resulting sorted sequences, we determine intervals so that there are $N/4d$ points on either sides of them. In each of these intervals, by another simple $O(N)$ scan we can determine the maximum of the widths of the windows (subintervals) containing exactly $\lfloor cdN^{1-1/d} \rfloor$ points. The maximum of these maxima yields a unique coordinate of the space and an interval whose bisector is the sought cut-plane. Note that, exclusive of presorting, the cut-plane can be determined in time linear in the size of S.

Thus, we have obtained a *sparse set S_{12} of cardinality $\leq cdN^{1-1/d}$* (with sparsity $c = 4 \cdot 3^{d-1}$ for the computed δ). We next solve Problem P.7 for S_{12}^*, the projection of S_{12} on the previously obtained cut-plane l. We shall use again the divide-and-conquer approach for this $(d-1)$-dimensional problem. Specifically, we find a cut-plane l', as guaranteed by Theorem 5.4, with given δ (as obtained in Step 2 of procedure CPAIRd), and c equal to the sparsity of S_{12}^*, just obtained. It follows that the point set in the δ-slice around the cut plane l' has size bounded by $(d-1) \cdot cM^{1-1/(d-1)}$, i.e., $o(M)$. In conclusion, the $(d-1)$-dimensional problem P.7 is solved by determining, in time $O(M)$, a cut-plane, followed by recursively solving the same problem on two subsets of sizes αM and $(1-\alpha)M$, and finally by solving a residual $(d-2)$-dimensional problem on a set size $o(M)$. This leads to the recurrence

$$F_{\delta,c}(M, d-1) = F_{\delta,c}\left(\frac{M}{4d}, d-1\right) + F_{\delta,c}\left(M\left(1 - \frac{1}{4d}\right), d-1\right) + O(M) \quad (5.3)$$
$$+ F_{\delta,c}(o(M), d-2).$$

This recurrence is readily solved as $F_{\delta,c}(M, d-1) = O(M \log M)$. This fact is interesting *per se* and is summarized in the following theorem.

Theorem 5.5. *All the pairs at distance less than δ in a sparse set of M points in E^d (with sparsity c for given δ) can be reported in time $O(M \log M)$.*

On the other hand, returning to the original closest-pair problem, M, the cardinality of S_{12}, is by construction $O(N^{1-1/d})$, i.e., it is $o(N/\log N)$. It follows that $F_{\delta,c}(M, d-1) = O(M \log M) = O(N)$. As a conclusion, recurrence relation (5.2) has solution $P(N, d) = O(N \log N)$. This computational work is of the same order as that of the initial presorting. So we have

Theorem 5.6. *The determination of a closest pair of points in a set of N points in E^d can be completed in time $\theta(N \log N)$, and this is optimal.*

5.5 The Locus Approach to Proximity Problems: The Voronoi Diagram

While the previous divide-and-conquer approach for the closest-pair problem is quite encouraging, it even fails to solve the ALL NEAREST NEIGHBORS problem, which would seem to be a simple extension. Indeed, if we try to set up the analogous recursion for ALL NEAREST NEIGHBORS, we find that the natural way of splitting the problem does not induce sparsity, and there is no apparent way of accomplishing the merge step in less than quadratic time. On the other hand, a valuable heuristic for designing geometric algorithms is to look at the defining loci and try to organize them into a data structure. In a two-dimensional formulation, we want to solve

PROBLEM P.8 (LOCI OF PROXIMITY). Given a set S of N points in the plane, for each point p_i in S what is the locus of points (x, y) in the plane that are closer to p_i than to any other point of S?

Note that, intuitively, the solution of the above problem is a partition of the plane into regions (each region being the locus of the points (x, y) closer to a point of S than to any other point of S). We also note that, if we know this partition, by searching it (i.e., by locating a query point q in a region of this partition), we could directly solve the NEAREST-NEIGHBOR SEARCH (Problem P.5). We shall now analyze the structure of this partition of the plane. Given two points, p_i and p_j, the set of points closer to p_i than to p_j is just the half-plane containing p_i that is defined by the perpendicular bisector of $\overline{p_i p_j}$. Let us denote this half-plane by $H(p_i, p_j)$. The locus of points closer to p_i than to any other point, which we denote by $V(i)$, is the *intersection of* $N - 1$ *half-planes*, and is a convex polygonal region (see Section 1.3.1) having no more than $N - 1$ sides, that is,

$$V(i) = \bigcap_{i \neq j} H(p_i, p_j).$$

$V(i)$ is called the *Voronoi polygon associated with* p_i. A Voronoi polygon is shown in Figure 5.16(a) [Rogers (1964)].[12]

[12] These polygons were first studied seriously by the emigré Russian mathematician G. Voronoi, who used them in a treatise on quadratic forms [Voronoi (1908)]. They are also called Dirichlet regions, Thiessen polygons, or Wigner-Seitz cells. Dan Hoey has suggested the more descriptive (and impartial) term "proximal polygons."