SIAM J. COMPUT.
Vol. 11, No. 4, November 1982


0097-5397/82/1104-0008 $01.00/0



# ON CONSTRUCTING MINIMUM SPANNING TREES IN $k$-DIMENSIONAL SPACES AND RELATED PROBLEMS*

ANDREW CHI-CHIH YAO†

**Abstract.** The problem of finding a minimum spanning tree connecting $n$ points in a $k$-dimensional space is discussed under three common distance metrics: Euclidean, rectilinear, and $L_\infty$. By employing a subroutine that solves the post office problem, we show that, for fixed $k \geqq 3$, such a minimum spanning tree can be found in time $O(n^{2-a(k)}(\log n)^{1-a(k)})$, where $a(k) = 2^{-(k+1)}$. The bound can be improved to $O((n \log n)^{1.8})$ for points in 3-dimensional Euclidean space. We also obtain $o(n^2)$ algorithms for finding a farthest pair in a set of $n$ points and for other related problems.

**Key words.** algorithm, minimum spanning tree, nearest neighbor, post office problem

## 1. Introduction.

Given an undirected graph with a weight assigned to each edge, a minimum spanning tree (MST) is a spanning tree whose edges have a minimum total weight among all spanning trees. The classical algorithms for finding MST were given by Dijkstra [7], Kruskal [13], Prim [14], and Sollin [4, p. 179]. It is well known (e.g., see Aho, Hopcroft and Ullman [1]) that, for a graph with $n$ vertices, an MST can be found in $O(n^2)$ time. (All time bounds discussed in this paper are for the worst-case behavior of algorithms.) For a sparse graph with $e$ edges and $n$ vertices, it was shown by Yao [16] that an MST can be found in time $O(e \log\log n)$. More studies of MST algorithms can also be found in Cheriton and Tarjan [6], Kerschenbaum and Van Slyke [11].

An interesting application of MST occurs in connection with hierarchical clustering analysis in pattern recognition (see, for example, Dude and Hart [9, Ch. 6], Zahn [20]). In this application, $n$ vertices $V = \{\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_n\}$ are given, each a $k$-tuple of numbers. The graph is understood to be a complete graph $G(V)$ on these $n$ vertices, with the weight on each edge $\{\tilde{v}_i, \tilde{v}_j\}$ being $d(\tilde{v}_i, \tilde{v}_j)$ where $d$ is a certain metric function computable from the components of $\tilde{v}_i$ and $\tilde{v}_j$. A simple way to find an MST in this case is to compute all the weights $d(\tilde{v}_i, \tilde{v}_j)$, and then use an $O(n^2)$ MST algorithm for general graphs. However, as there are only $kn$ input parameters, it is interesting to find out if there are algorithms which take only $o(n^2)$ time. Several empirically good algorithms were proposed in Bentley and Friedman [2], where a list of references to other applications of finding MST in $k$-dimensional spaces can also be found. Shamos and Hoey [16] gave an $O(n \log n)$ algorithm for $n$ points in the plane ($k = 2$) with Euclidean metric. No algorithm, however, is known to have a guaranteed bound of $o(n^2)$ when $k \geqq 3$.

In this paper, we consider three common metrics in $k$-dimensional spaces, namely, the rectilinear ($L_1$), the Euclidean ($L_2$), and the $L_\infty$ metric. We use $E_p^k$ ($p = 1, 2, \infty$) to denote the space of all $k$-tuples of real numbers with the $L_p$-metric, i.e., the distance between two points $\tilde{x}$ and $\tilde{y}$ is given by $d_p(\tilde{x}, \tilde{y}) = (\sum_{i=1}^{k} |x_i - y_i|^p)^{1/p}$. (It is agreed that $d_\infty(\tilde{x}, \tilde{y}) = \max_i |x_i - y_i|$.) We give new algorithms which construct, for a given set $V$ of $n$ points in $E_p^k$, an MST for the associated complete graph $G(V)$. The algorithms work in time $O(n^{2-a(k)}(\log n)^{1-a(k)})$, where $a(k) = 2^{-(k+1)}$ for any fixed $k \geqq 3$. Fast algorithms for related geometric problems are also given using similar techniques.

* Received by the editors November 3, 1980. This research was supported in part by National Science Foundation under grant MCS 72-03752 A03.

† Computer Science Department, Stanford University, Stanford, California 94305.

The main results of this paper are summarized in the following theorem. Sections 2–5 are devoted to a proof of it.

THEOREM 1. *Let $k \geqq 3$ be a fixed integer, $a(k) = 2^{-(k+1)}$, and all points to be considered are in $E_p^k$ with $p \in \{1, 2, \infty\}$. Then each of the following problems can be solved in time $O(n^{2-a(k)}(\log n)^{1-a(k)})$. For the case when $k = 3$ and $p = 2$, the bound can be improved to $O((n \log n)^{1.8})$.*

| | |
|---|---|
| MST-*problem* | *Let $V$ be a set of $n$ points; find a minimum spanning tree on $V$.* |
| NFN-*problem* | (*nearest foreign neighbor*). *Let $V_1, V_2, \cdots, V_l$ be disjoint sets of points, $V = \cup_i V_i$, and $\|V\| = n$. For each $V_i$ and every $\tilde{x} \in V_i$, find a $\tilde{y} \in V - V_i$ such that $d_p(\tilde{x}, \tilde{y}) = \min\{d_p(\tilde{x}, \tilde{z}) \mid \tilde{z} \in V - V_i\}$.* |
| GN-*problem* | (*geographic neighbor*). *Let $V$ be a set of $n$ points. For any $\tilde{x} \in V$, let $N(\tilde{x}) = \{\tilde{v} \mid v_i \geqq x_i$ for all $1 \leqq i \leqq k, \tilde{v} \neq \tilde{x}, \tilde{v} \in V\}$. For each $\tilde{x} \in V$, find a $\tilde{y} \in N(\tilde{x})$ such that $d_p(\tilde{x}, \tilde{y}) = \min\{d_p(\tilde{x}, \tilde{v}) \mid \tilde{v} \in N(x)\}$ if $N(\tilde{x}) \neq \varnothing$.* |
| AFP-*problem* | (*all farthest points*) [3]. *Let $V$ be a set of $n$ points. For each $\tilde{x} \in V$, find a $\tilde{y} \in V$ such that $d_p(\tilde{x}, \tilde{y}) = \max\{d_p(\tilde{x}, \tilde{v}) \mid \tilde{v} \in V\}$.* |
| FP-*problem* | (*farthest pair*) [3]. *Let $V$ be a set of $n$ points; find $\tilde{x}, \tilde{y} \in V$ such that $d_p(\tilde{x}, \tilde{y}) = \max\{d_p(\tilde{u}, \tilde{v}) \mid \tilde{u}, \tilde{v} \in V\}$.* |

In § 6, we briefly describe, for the $L_2$ and the $L_\infty$ metric, how to obtain $o(kn^2)$ algorithms when $k$ is allowed to vary with $n$.

A remark on the model of computation: We assume a random access machine with arithmetic on real numbers, and charge uniform cost for all access and arithmetic operations [1]. In this paper, we often carry out computations of $d_p(\tilde{x}, \tilde{y})$, which involves an apparent square root operation when $p = 2$. However, since our construction of MST depends only on the linear ordering among the edge weights, we can replace $d_p(\tilde{x}, \tilde{y})$ throughout by some monotone function of $d_p(\tilde{x}, \tilde{y})$. In particular, $d_2(\tilde{x}, \tilde{y})$ may be replaced by $(d_2(\tilde{x}, \tilde{y}))^2 = \sum (x_i - y_i)^2$ everywhere to produce a valid algorithm without square root operations. We shall, however, retain the original form of the algorithm for clarity and for consistency with the cases $p = 1, \infty$.

**2. The post-office problem and its applications.** In this section we review solutions to the post-office problem, and show how it can be used to prove Theorem 1 for the AFP, FP and NFN problems.

The *post-office problem* can be stated as follows. Given a set of $n$ points $\tilde{v}_1, \tilde{v}_2, \cdots, \tilde{v}_n$ in $E_p^k$, we wish to preprocess them so that any subsequent query of the following form can be answered quickly:

*nearest-point query*. Given a point $\tilde{x}$, find a nearest $\tilde{v}_i$ to $\tilde{x}$ (i.e., $d_p(\tilde{x}, \tilde{v}_i) \leqq d_p(\tilde{x}, \tilde{v}_j)$ for all $j$).

This problem was mentioned in Knuth [12] for the case of points in the Euclidean plane ($k = p = 2$). For this special case, several solutions were given by Dobkin and Lipton [8] and Shamos [15]. For example, it is known that with an $O(n^2)$-time preprocessing, any nearest-point query can be answered in $O(\log n)$ time [15]. A solution for the $k$-dimensional Euclidean space was given in Dobkin and Lipton [8], where it was shown that it is possible to preprocess $n$ points such that any subsequent nearest-point query can be answered in $O(2^k \log n)$ time. Their technique is quite general, and applies equally well if we wish to answer "farthest-point" queries—Given $\tilde{x}$, find a farthest $\tilde{v}_i$ to $\tilde{x}$—instead of nearest-point queries. The preprocessing procedure was not discussed in great detail in [8]. A detailed study [19] gives the following result.

DEFINITION. We shall use $b(k) = 2^{k+1}$, and $a(k) = b(k)^{-1} = 2^{-(k+1)}$.

LEMMA 2.1. *Let $k \geqq 3$ be a fixed integer, and $p \in \{1, 2, \infty\}$. There is an algorithm which preprocesses $n$ points in $E_p^k$ in time $O(n^{b(k)})$ such that each subsequent nearest-point query can be answered in $O(\log n)$ time. In the special case $k = 3, p = 2$, the preprocessing time can be improved to $O(n^5 \log n)$ with a query response-time $O((\log n)^2)$. The preceding statements remain true if the farthest-point query is used in place of the nearest-point query.*

We shall now demonstrate the use of Lemma 2.1 by applying it to solve the MST problem in a special case. It also gives us some insight into the connection between MST and some typical nearest-neighbor problem [3], [16].

Consider the case when $V$ consists of two widely separated clusters $A$ and $B$. For definiteness, assume that $d_p(A, B) > n \cdot (\text{diam}\,(A) + \text{diam}\,(B))$[1]. In this case any MST on $V$ consists of the union of an MST for $A$ and an MST for $B$, plus a shortest edge between $A$ and $B$. Thus, to be able to solve the MST problem efficiently, we have to be able to solve the following problem efficiently:

*Problem* RMST. Given two well-separated sets $A$ and $B$ in $E_p^k$, with $|A| = |B| = n$, find a shortest edge between $A$ and $B$.

This problem looks very similar to the problem of finding the closest pair in a set, which has an $O(n \log n)$-time algorithm. However, there does not seem to be any simple divide-and-conquer $o(n^2)$ solution. We shall presently give an $o(n^2)$-time algorithm employing the post-office problem as a subroutine.

Consider the following algorithm.

(S1) Divide $B$ into $r = \lceil n/q \rceil$ sets $B_1, B_2, \cdots, B_r$, each with at most $q$ points ($q$ to be determined).
(S2) For each $1 \leqq i \leqq r$, preprocess $B_i$ for nearest-point queries as in Lemma 2.1.
(S3) For each $\tilde{x} \in A$ and each $1 \leqq i \leqq r$, find a point $\tilde{y}(\tilde{x}, i) \in B_i$ that is nearest to $\tilde{x}$ among all points in $B_i$.
(S4) For each $\tilde{x} \in A$, find a $\tilde{z}(\tilde{x}) \in B$ nearest to $x$ by comparing $\tilde{y}(\tilde{x}, i)$ for all $1 \leqq i \leqq r$.
(S5) Find a shortest such edge $\{\tilde{x}, \tilde{z}(\tilde{x})\}$.

The time taken is dominated by (S2) and (S3), i.e.,

$$O(r \cdot q^{b(k)} + nr \log q).$$

Choosing $q = (\log n)^{(b(k))^{-1}}$ gives the time $O(n(n \log n)^{1-(b(k))^{-1}})$. Thus, we have found an algorithm that solves RMST in time $O(n^{2-a(k)}(\log n)^{1-a(k)})$. For the case $k = 3$ and $p = 2$, one can choose $q = (n \log n)^{1/5}$ to obtain an $O((n \log n)^{1.8})$ algorithm.

We wish to make two observations concerning the above procedure. Firstly, the AFP- and FP-problems can be solved with the same time bounds by very similar procedures (employing farthest-point queries and preprocessing, of course). We will thus consider that Theorem 1 has been proved for these problems. Secondly, the RMST-problem is a type of nearest-neighbor problem with some restrictions on the "legal" neighbors. It is reasonable to expect more such problems can be solved with similar techniques. The NFN- and GN-problems are problems of this type, and we will see that their efficient solutions enable the MST problem to be solved efficiently. We shall give a fast algorithm for NFN-problems presently, leaving the more involved proof of Theorem 1 for MST and GN to the later sections.

[1] We use the notation $d_p(A, B) = \min\{d_p(\tilde{u}, \tilde{v}) \mid \tilde{u} \in A, \tilde{v} \in B\}$, $d_p(\tilde{u}, S) = \min\{d_p(\tilde{u}, \tilde{v}) \mid \tilde{v} \in S\}$, and $\text{diam}\,(S) = \max\{d_p(\tilde{u}, \tilde{v}) \mid \tilde{u}, \tilde{v} \in S\}$.

We are given disjoint sets $V_1, V_2, \cdots, V_l$ with a total of $n$ points in $V = \bigcup_i V_i$. For a point $\tilde{x} \in V_i$, every point $\tilde{y} \in V - V_i$ is a *foreign neighbor* of $\tilde{x}$. Let $q = \lceil (n \log n)^{a(k)} \rceil$; call a set $V_i$ *small* if $|V_i| < q$, and *large* if $|V_i| \geqq q$. We partition $V$ into $r = O(n/q)$ parts $B_1, B_2, \cdots, B_r$, where each part (call it a *block*) either is the union of several small $V_i$ or is totally contained in some large $V_i$. Furthermore, each part contains at most $2q$ points, and except possibly for $B_r$, at least $q$ points. The above partition can be accomplished in $O(n)$ time by breaking each large $V_i$ into several blocks and grouping small $V_i$ into blocks of appropriate sizes. We now preprocess each block $B_i$ so that, for any query point $\tilde{x}$, a point nearest to $\tilde{x}$ in $B_i$ can be found in $O(\log q)$ time. According to Lemma 2.1, this preprocessing can be accomplished in time $O(rq^{b(k)})$ for all blocks $B_i$. We are now ready to find, for each point $\tilde{x} \in V$, a nearest foreign neighbor $\tilde{y}$, i.e., $d_p(\tilde{x}, \tilde{y}) = \min \{d_p(\tilde{x}, \tilde{z}) \mid \tilde{z} \in V - V_i\}$, when $\tilde{x} \in V_i$. Assume that $\tilde{x} \in V_i$ and $\tilde{x} \in B_t$. Let us find, for each block $B_j$ that is disjoint from $V_i$, a point $\tilde{z}(\tilde{x}, j)$ nearest to $\tilde{x}$ among all points in $B_j$. Then we find a nearest foreign neighbor $\tilde{y}$ from the points $\tilde{z}(\tilde{x}, j)$ and points in $B_t - V_i$ by computing and comparing their distances to $\tilde{x}$. The running time for finding $\tilde{y}$, for each $\tilde{x}$, is thus $O(r \log q + (r+q))$. In summary, the total running time of the above procedure for NFN is $O(n + rq^{b(k)} + nr \log q + nq)$, which is $O(n^{2-a(k)}(\log n)^{1-a(k)})$. As before, an $O((n \log n)^{1.8})$ algorithm can be obtained for the case $k = 3$ and $p = 2$.

This proves Theorem 1 for the NFN-problem. An interesting connection exists between MST- and NFN-problems. In fact, in Sollin's algorithm[4, p. 179], an MST can be found essentially by solving NFN-problems $O(\log n)$ times. Thus, we have shown that an MST can be found in $\log n \times O(n^{2-a(k)}(\log n)^{1-a(k)})$-time. The $\log n$ factor can be avoided by reducing MST to a generalized version of the GN-problem, which can be solved in time $O(n^{2-a(k)}(\log n)^{1-a(k)})$. The proof requires additional techniques beyond the simple application of post-office problems to small parts of $V$. We shall illustrate the ideas for two dimensions in the next section, and complete the proof in §§ 4 and 5.

**3. An illustration in two dimensions.** We illustrate the ideas of our MST-algorithms with an informal description for 2-dimensional Euclidean space. Let us first consider a special type of nearest-neighbor problem. Let $\tilde{p}$ be any point in the plane. We divide the plane into eight regions relative to $\tilde{p}$ as shown in Fig. 1. The regions are formed by four lines passing through $\tilde{p}$ and having angles of 0°, 45°, 90°, and 135°, respectively, with the $x$-axis. We number the regions counterclockwise as shown in Fig. 1, and use $R_l(\tilde{p})$ to denote the set of points in the $l$th region (including its boundary), for $1 \leqq l \leqq 8$.

LEMMA 3.1. *If $\tilde{q}$ and $\tilde{q}'$ are two points in $R_l(\tilde{p})$ for some $l$, then $d_2(\tilde{q}, \tilde{q}') < \max \{d_2(\tilde{p}, \tilde{q}), d_2(\tilde{p}, \tilde{q}')\}$.*

*Proof.* Consider the triangle $\tilde{p}\tilde{q}\tilde{q}'$ (see Fig. 1). Since $\angle \tilde{q}\tilde{p}\tilde{q}' \leqq 45° < \pi/3$, its opposite side $\tilde{q}\tilde{q}'$ cannot be the longest side of the triangle. □

Let $V$ be a set of $n$ distinct points in the plane. For each point $\tilde{v} \in V$, let $N_l(\tilde{v})$ be those points of $V$, excluding $\tilde{v}$ itself, that are in the $l$th region relative to $\tilde{v}$. That is,

$$N_l(\tilde{v}) = V \cap R_l(\tilde{v}) - \{\tilde{v}\} \quad \text{for } 1 \leqq l \leqq 8.$$

A point $\tilde{u}$ in $N_l(\tilde{v})$ is said to be a *nearest neighbor to $\tilde{v}$ in the $l$th region* if $d_2(\tilde{v}, \tilde{u}) = \min \{d_2(\tilde{v}, \tilde{w}) \mid \tilde{w} \in N_l(\tilde{v})\}$. Note that such a nearest neighbor does not exist if $N_l(\tilde{v}) = \varnothing$, and may not be unique when it exists. Now, consider the following computational problem.

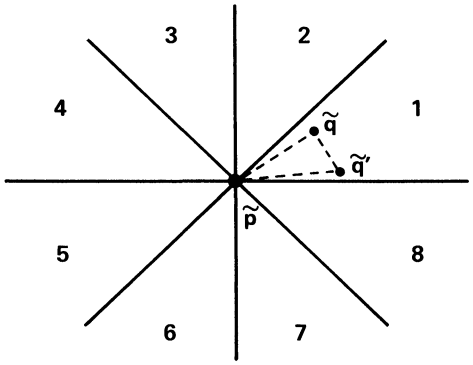


FIG. 1. *Regions $R_l(\tilde{p})$ for $1 \leqq l \leqq 8$.*

*The eight-neighbors problem* (ENP). Given a set $V$ of $n$ points in the plane, find for each $\tilde{v} \in V$ and $1 \leqq l \leqq 8$ a nearest neighbor to $\tilde{v}$ in the $l$th region, if it exists.

We first show that, once the eight-neighbors problem is solved for $V$, it takes very little extra effort to find an MST on $V$. To see this, we form $E$, the set of edges defined by:

$$E = \{\{\tilde{v}, \tilde{u}\} \mid \tilde{v} \in V \text{ and } \tilde{u} \text{ is a nearest neighbor to } \tilde{v} \text{ selected by ENP}\}.$$

We assert that the set of edges $E$ contains an MST on $V$. As $E$ contains at most $8n$ edges, we can then construct an MST for the sparse graph $(V, E)$ in $O(n \log\log n)$ steps [17], a very small cost.

THEOREM 3.2. *The set of edges $E$ contains an MST on $V$.*

*Proof.* Let $T$ be a set of edges that form an MST on $V$. We will show that, for any edge $\{\tilde{v}, \tilde{w}\}$ that is in $T$ but not in $E$, we can replace $\{\tilde{v}, \tilde{w}\}$ by an edge in $E$ and still maintain an MST. This would prove the theorem since we can perform this operation on $T$ repeatedly until all edges in $T$ are from $E$.

Let $\{\tilde{v}, \tilde{w}\}$ be an edge in $T - E$. Assume $\tilde{w} \in R_l(\tilde{v})$. Then $N_l \neq \varnothing$, and there is a nearest neighbor $\tilde{u}$ to $\tilde{v}$ in $N_l(\tilde{v})$ such that $\{\tilde{v}, \tilde{u}\} \in E$. Clearly $\tilde{u} \neq \tilde{w}$ and $d_2(\tilde{v}, \tilde{u}) \leqq d_2(\tilde{v}, \tilde{w})$. Let us delete $\{\tilde{v}, \tilde{w}\}$ from $T$. Then $T$ is separated into two disjoint subtrees with $\tilde{v}$ and $\tilde{w}$ belonging to different components. Now, $\tilde{u}$ and $\tilde{w}$ must be in the same component. For if they were not, $\{\tilde{u}, \tilde{w}\}$ would be a shorter connecting edge for the two subtrees than $\{\tilde{v}, \tilde{w}\}$ by Lemma 3.1, contradicting the fact that $T$ is an MST. Therefore $\tilde{u}$ is in the same subtree as $\tilde{w}$, and adding the edge $\{\tilde{v}, \tilde{u}\}$ to $T - \{\tilde{v}, \tilde{w}\}$ results in a spanning tree with total weight no greater than that of $T$. □

We now proceed to solve the eight-neighbors problem. We will find a nearest neighbor to each point in the first region. The procedure can be simply adapted to find nearest neighbors in the $l$th region for other $l$. As demonstrated earlier, the MST problem can be thus solved in a total of $8 \cdot f(n) + O(n \log\log n)$ steps, if the first-region nearest neighbors can be found in $f(n)$ steps.

To study the first regions, it is convenient to tilt the $y$-axis by 45° clockwise (see Fig. 2). That is, transform the coordinates $(x_1, x_2)$ of a point $v$ into $(x_1', x_2')$, defined by

$$\begin{pmatrix} x_1' \\ x_2' \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ 0 & \sqrt{2} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

In the new coordinates, a point $\tilde{u} = (u_1', u_2')$ is in the first region relative to $v = (v_1', v_2')$ if and only if $(u_1' \geqq v_1') \wedge (u_2' \geqq v_2')$.

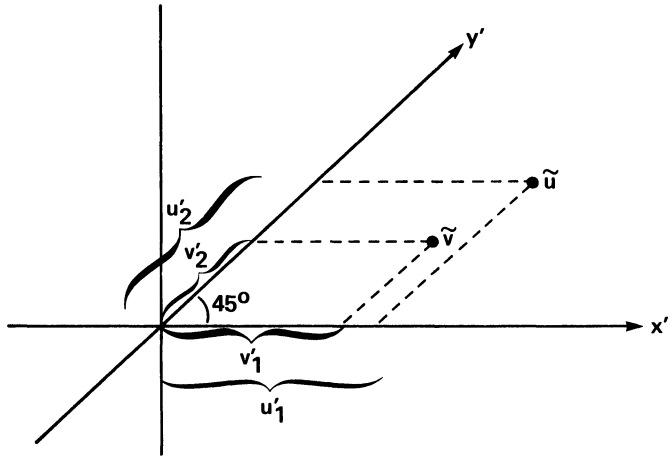


FIG. 2. *New coordinate system.*

For simplicity we assume that all the $2n$ coordinates $x'_1$, $x'_2$ of points $\tilde{x} \in V$ are distinct numbers. This restriction shall be removed in the general algorithm in § 3. Let us first sort the points according to their first coordinates $x'_1$, and divide them into $s = (n/q)^{1/2}$ consecutive groups each with $\approx qs$ points (Fig. 3), $q$ to be determined later. Then for each of these $s$ groups we sort the points in ascending order of the coordinates $x'_2$, and divide them into $s$ consecutive groups with $\approx q$ points each (Fig. 4).

The set $V$ is thus divided into $s^2$ "cells". For any $\tilde{v} \in V$, the cells can be classified into three classes by their position relative to $\tilde{v}$: class 1, cells all of whose points are in $N_1(\tilde{v})$; class 2, cells with no points in $N_1(\tilde{v})$; and class 3, the remaining cells. A useful observation is that the number of cells in class 3 is at most $2 \times s$. This can be understood as follows: If we draw a horizontal and a vertical line through $v$, only those cells that are "hit" can be in class 3, and there are at most $2 \times s$ of them. We can now try to find a nearest neighbor for $\tilde{v}$ in $N_1(\tilde{v})$ using the following strategy: we examine each cell in turn for cells in class 3, and compute $d_2(\tilde{v}, \tilde{u})$ for all $\tilde{u}$ in the

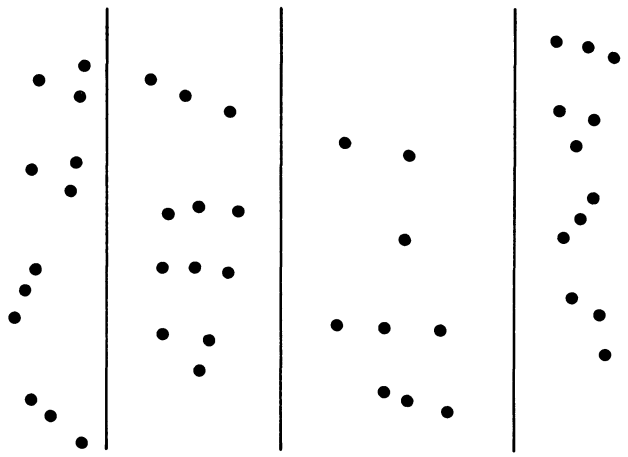

FIG. 3. *Division of points into s groups according to values of* $x'_1$.

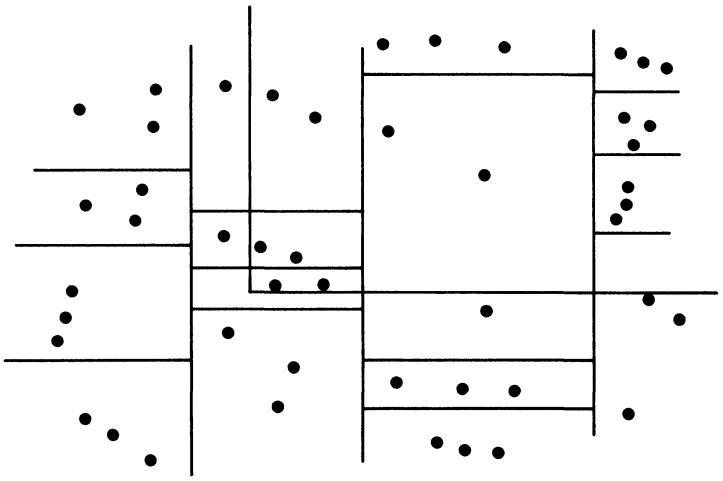

FIG. 4. *Completing the division of V into $s^2$ cells.*

cell; for a cell in class 2, we ignore it; for a cell $C$ in class 1, we compute $\tilde{u}$ and $d_2(\tilde{v}, \tilde{u})$ defined by $d_2(\tilde{v}, \tilde{u}) = \min\{d_2(\tilde{v}, \tilde{x}) | \tilde{x} \in C\}$. A nearest point can now be found by selecting the point $\tilde{u}$ with minimum $d_2(\tilde{v}, \tilde{u})$ from the preceding calculations. The cost is $O(2s \cdot q + \#$ of class 1 cells $\times a) = O(2sq + s^2 a) = O(n/s + 2an/q)$, where $a$ is the cost of computing $d_2(v_i, C)$ for a cell $C$ of $q$ points. If we have to compute $d_2(\tilde{v}, \tilde{u})$ for each $\tilde{u} \in C$, then $a = O(q)$, and the total cost would be $O(n)$, and we have not made any progress. However, we know from the post-office problem that we can lower $a$ to $\log q$ if we are willing to preprocess the set $C$ (in $O(q^2)$ time). So let us do the following: (i) preprocess every cell $C$ to facilitate the computing of $d_2(\tilde{v}, C)$ (cost $O((n/q) \cdot q^2) = O(nq)$); (ii) for each $\tilde{v}$, compute the nearest neighbor in the above manner in time $O(n/s + (n/q) \log q)$. The total cost is then $O(nq + n^2/s + (n^2/q) \log q)$. Take $q = n^{1/3}$ and obtain an algorithm that runs in time $O(n^{5/3} \log n)$. This gives an $o(n^2)$ algorithm for finding an MST in two dimensions. We shall generalize the ideas to general $k$.

**4. Reduction of MST to a general GN-problem.** We shall prove Theorem 1 for the MST- and GN-problems in this and the next sections. Without loss of generality, we shall assume that the $n$ given points in $V$ are all distinct.

In this section we reduce the finding of MST in $E_p^k$ to a version of the geographic-neighbor problem. We assume that $p \in \{1, 2, \infty\}$ throughout the rest of the paper.

We make $E_p^k$ a vector space by defining $\tilde{x} + \tilde{y} = (x_1 + y_1, x_2 + y_2, \cdots, x_k + y_k)$ and $c\tilde{x} = (cx_1, cx_2, \cdots, cx_k)$, where $c$ is any real number and $x_i$, $y_i$ are the components of $\tilde{x}$ and $\tilde{y}$. We shall refer to any element of $E_p^k$ as a *point* or a *vector*. The $j$th component of a vector $\tilde{z}$ will be denoted as $z_i$ without further explanation. The *inner product* of two vectors $\tilde{x}$ and $\tilde{y}$ is $\tilde{x} \cdot \tilde{y} = \sum_{i=1}^k x_i y_i$, and the *norm* of $\tilde{x}$ is $\|\tilde{x}\| = (\tilde{x} \cdot \tilde{x})^{1/2}$. A *unit vector* $\tilde{x}$ is a vector with $\|\tilde{x}\| = 1$. Notice that all these definitions are independent of $p$.

Vectors $\tilde{b}_1, \tilde{b}_2, \cdots, \tilde{b}_j$ are *linearly independent* if $\sum_{i=1}^j \lambda_i \tilde{b}_i = 0$ implies all $\lambda_i = 0$. A set of $k$ linearly independent vectors in $E_p^k$ is called a *basis* (of $E_p^k$). Let $B = \{\tilde{b}_1, \tilde{b}_2, \cdots, \tilde{b}_k\}$ be a basis of $E_p^k$. The *convex cone* of $B$ is $\text{Conv}(B) = \{\sum_{i=1}^k \lambda_i \tilde{b}_i | \lambda_i \geqq 0 \text{ for all } i\}$. For any $\tilde{x} \in E_p^k$, *the region B of* $\tilde{x}$ is defined as

$$R(B; \tilde{x}) = \{\tilde{y} | \tilde{y} - \tilde{x} \in \text{Conv}(B)\}.$$

Let $V$ be a set of $n$ distinct vectors in $E_p^k$. Denote by $N(B, \tilde{v})$ the set $V \cap \{\tilde{u} \mid \tilde{u} \in R(B; \tilde{v}) - \{\tilde{v}\}\}$, for each $\tilde{v} \in V$. We shall say that $\tilde{w}$ is a *geographic neighbor to* $\tilde{v}$ *in region* $B$ if $\tilde{w} \in N(B; v)$ and $d_p(\tilde{w}, \tilde{v}) \leqq d_p(\tilde{u}, \tilde{v})$ for all $\tilde{u} \in N(B; \tilde{v})$.

*The* GGN-*problem* (*general geographic neighbor*). Given a basis $B$ and a set $V$ of $n$ distinct vectors in $E_p^k$, find, for each $\tilde{v} \in V$, a geographic neighbor to $\tilde{v}$ in region $B$, if one exists.

Notice that this reduces to the GN-problem when $B = \{\tilde{b}_1, \tilde{b}_2, \cdots, \tilde{b}_k\}$ with $b_{ij} = \delta_{ij}$. The rest of this section is devoted to showing the following theorem, which states that, if there is a fast algorithm to solve the GGN-problem, then one can solve the MST-problem efficiently.

THEOREM 4.1. *Let $k \geqq 2$ be a fixed integer. Suppose there is an algorithm that solves the* GGN-*problem for $n$ given points in $E_p^k$ in at most $f(n)$ steps. Then a minimum spanning tree for $n$ points in $E_p^k$ can be found in $O(f(n)+n \log\log n)$ steps.*

Define the *angle* between two nonzero vectors $\tilde{x}$ and $\tilde{y}$ as $\Theta(\tilde{x}, \tilde{y}) = \cos^{-1}(\tilde{x} \cdot \tilde{y}/\|\tilde{x}\| \cdot \|\tilde{y}\|)$, $0 \leqq \Theta(\tilde{x}, \tilde{y}) \leqq \pi$. For any basis $B$ of $E_p^k$, the *angular diameter* of $B$ is defined by $\text{Ang}(B) = \sup\{\Theta(\tilde{x}, \tilde{y}) \mid \tilde{x}, \tilde{y} \in \text{Conv}(B)\}$. It can be shown that $\text{Ang}(B) = \max\{\Theta(\tilde{b}_i, \tilde{b}_j) \mid \tilde{b}_i, \tilde{b}_j \in B\}$, although we shall not use that fact.

Let $\mathcal{B}$ be a finite family of basis of $E_p^k$. We call $\mathcal{B}$ a *frame* if $\bigcup_{B \in \mathcal{B}} \text{Conv}(B) = E_p^k$. The *angular diameter* of a frame $\mathcal{B}$ is given by $\text{Ang}(\mathcal{B}) = \max\{\text{Ang}(B) \mid B \in \mathcal{B}\}$. For example, let $\tilde{b}_1 = (1, 0)$, $\tilde{b}_2 = (-1, 1)$, $\tilde{b}_3 = (0, -1)$, $\tilde{b}_4 = (-\frac{1}{2}, -1)$ as shown in Fig. 5, then

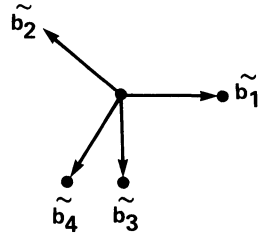


FIG. 5. *Illustration of "basis" and "frame".*

$B_1 = \{\tilde{b}_1, \tilde{b}_2\}$, $B_2 = \{\tilde{b}_2, \tilde{b}_3\}$, $B_3 = \{\tilde{b}_4, \tilde{b}_1\}$ are bases of $E_p^2$, and $\mathcal{B} = \{B_1, B_2, B_3\}$ a frame; $\Theta(B_1) = \Theta(B_2) = 3\pi/4$, $\Theta(B_3) = 2\pi/3$, and $\Theta(\mathcal{B}) = 3\pi/4$.

Intuitively, the convex cone of a basis $B$ has a "narrow" angular coverage if $\text{Ang}(B)$ is small. The following result asserts that a frame exists in which every basis is narrow, and such a frame can be constructed.

LEMMA 4.2. *For any $0 < \psi < \pi$, one can construct in finite steps a frame $\mathcal{B}$ of $E_p^k$ such that* $\text{Ang}(\mathcal{B}) < \psi$.

*Proof.* See Appendix. □

We consider the following MST-algorithm. Let us construct a frame $\mathcal{B}$ of $E_p^k$ such that $\text{Ang}(B) < \sin^{-1}(\frac{1}{2}k^{-(1/2+1/p)})$. Next, for each $B \in \mathcal{B}$, we solve the GGN-problem—for each $\tilde{v} \in V$, find a geographic neighbor $\tilde{u}$ to $\tilde{v}$ in region $B$ if it exists—and form the set $E(B)$, the collection of all such edges $\{\tilde{u}, \tilde{v}\}$. Clearly, $|\bigcup_{B \in \mathcal{B}} E(B)| \leqq n \cdot |\mathcal{B}| = O(n)$. We now claim that $\bigcup_{B \in \mathcal{B}} E(B)$ contains an MST on $V$. If this is true, then we can find an MST in an additional $O(n \log\log n)$ steps. The total time taken by the MST algorithm is then $O(f(n) + n \log\log n)$. It remains to prove the following result.

LEMMA 4.3. $\bigcup_{B \in \mathcal{B}} E(B)$ *contains an* MST *on* $V$.

*Proof.* The proof is almost identical to the proof of Theorem 3.2, except that we need to establish the next lemma. □.

LEMMA 4.4. *Let* $\tilde{x}$, $\tilde{y}$, $\tilde{z}$ *in* $E_p^k$ *satisfy* $\Theta(\tilde{x}-\tilde{z}, \tilde{y}-z) < \sin^{-1}(\frac{1}{2}k^{-(1/2+1/p)})$, *then* $d_p(\tilde{x}, \tilde{y}) < \max\{d_p(\tilde{y}, \tilde{z}), d_p(\tilde{x}, \tilde{z})\}$.

*Proof.* Use $\alpha$, $\beta$, $\gamma$ to denote angles as shown in Fig. 6. By assumption,

$$\sin\alpha < \tfrac{1}{2}k^{-(1/2+1/p)}. \tag{1}$$

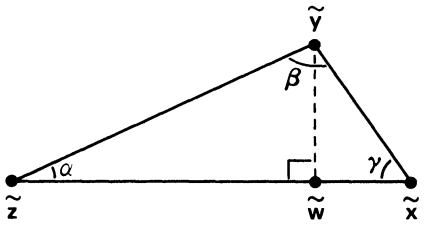


FIG. 6. *Illustration for the proof of Lemma* 4.4.

Without loss of generality, assume that $\alpha+\beta > \pi/2$. Let $\tilde{w}$ be the projection of $\tilde{y}$ on the segment from $\tilde{z}$ to $\tilde{x}$. By the triangle inequality satisfied by metric $d_p$, we have

$$d_p(\tilde{z}, \tilde{w}) + d_p(\tilde{w}, \tilde{y}) \geqq d_p(\tilde{y}, \tilde{z}),$$
$$d_p(\tilde{x}, \tilde{w}) + d_p(\tilde{w}, \tilde{y}) \geqq d_p(\tilde{x}, \tilde{y}).$$

Thus,

$$d_p(\tilde{z}, \tilde{w}) + d_p(\tilde{x}, \tilde{w}) \geqq d_p(\tilde{x}, \tilde{y}) + (d_p(\tilde{y}, \tilde{z}) - 2d_p(\tilde{w}, \tilde{y})). \tag{2}$$

But since $\tilde{w}$ is on the segment $\tilde{z}$ to $\tilde{x}$, we have $d_p(\tilde{x}, \tilde{z}) = d_p(\tilde{z}, \tilde{w}) + d_p(\tilde{x}, \tilde{w})$. Therefore, if we can further show that

$$d_p(\tilde{y}, \tilde{z}) - 2d_p(\tilde{w}, \tilde{y}) > 0, \tag{3}$$

then (2) implies $d_p(\tilde{x}, \tilde{z}) > d_p(\tilde{x}, \tilde{y})$, proving the lemma.

To prove formula (3), we notice that for any positive $l$, and $\tilde{u}$, $\tilde{v}$ in $E_l^k$,

$$k^{1/l} \max_i |\tilde{u}_i - \tilde{v}_i| \geqq d_l(\tilde{u}, \tilde{v}) \geqq \max_i |\tilde{u}_i - \tilde{v}_i|. \tag{4}$$

This leads to

$$k^{1/p}\|\tilde{u}-\tilde{v}\| \geqq d_p(\tilde{u}, \tilde{v}) \geqq k^{-1/2}\|\tilde{u}-\tilde{v}\|. \tag{5}$$

In particular,

$$d_p(\tilde{y}, \tilde{w}) \leqq k^{1/p}\|\tilde{y}-\tilde{w}\|,$$
$$d_p(\tilde{y}, \tilde{z}) \geqq k^{-1/2}\|\tilde{y}-\tilde{z}\|. \tag{6}$$

Now, clearly by (1),

$$\|\tilde{y}-\tilde{w}\| = (\sin\alpha)\|\tilde{y}-\tilde{z}\| < \tfrac{1}{2}k^{-(1/2+1/p)}\|\tilde{y}-\tilde{z}\|. \tag{7}$$

Formula (3) follows from (6) and (7). □

**5. An algorithm for the general geographic neighbor problem.**

**5.1. An outline.** As shown in the preceding section, the MST-problem can be reduced to the GGN-problem, and the GN-problem is a special case of the GGN-

problem. In this section, we shall give an asymptotically fast algorithm for the GGN-problem, which completes the proof of Theorem 1.

Given a basis $B$ and a set $V$ of $n$ points in $E_p^k$, the algorithm works in two phases.

*Preprocessing phase.* (A) Partition $V$ in $O(kn \log n)$ steps into $r = \lceil n/q \rceil$ subsets $V_1, V_2, \cdots, V_r$, each with at most $q$ points ($q$ to be determined later). The division will be such that, for any $\tilde{x} \in E_p^k$, all but a fraction $r^{-1/k}$ of the subsets $V_j$ have the property that the entire set $V_j$ is either in region $B$ of $\tilde{x}$ or outside of region $B$.

(B) Preprocess each $V_j$ in $O(q^{b(k)})$ steps such that, for any new point $\tilde{x} \in E^k$, a nearest point $\tilde{u}$ in $V_j$ can be found in $O(\log q)$ steps.

*Finishing phase.* (C) For each $\tilde{v} \in V$, we find a geographic neighbor in region $B$ as follows. We examine the $r$ sets $V_1, V_2, \cdots, V_r$ in turn. For each $V_j$, we perform a test which puts $V_j$ into one of the three categories. A category-1 $V_j$ has all its points in region $B$ of $\tilde{v}$, a category-2 $V_j$ has all its points outside of region $B$. The nature of a category-3 $V_j$ is unimportant, except that there are at most $r^{1-k^{-1}}$ $V_j$ in this category; we consider the $V_j$ that contains $\tilde{v}$ itself to be of category 3 independent of the above division. As we shall see later, the test will be easy to carry out, in fact in $O(k)$ time per test. For a category-1 $V_j$, we find a nearest $\tilde{w}$ in $V_j$ in $O(\log q)$ time. For a category-2 $V_j$, nothing need be done. For a category-3 $V_j$, we find a nearest $\tilde{w}(\neq \tilde{v}) \in V_j$ in region $B$, if it exists, by finding all the $\tilde{z} \in V_j$ that are in region $B$ and computing and comparing $d_p(\tilde{z}, \tilde{v})$ for all such $\tilde{z}$. Call $\tilde{w}$ a candidate from $V_j$. After all the $V_j$ have been so processed, we compare $d_p(\tilde{w}, \tilde{v})$ for all the candidates $\tilde{w}$ obtained (at most $r$ of them), and find a nearest one $\tilde{u}$ to $\tilde{v}$. This $\tilde{u}$ is the geographic neighbor we seek for $\tilde{v}$. Return "nonexistent" if no candidate $\tilde{w}$ exists from any $V_j$.

In the above description, three points need further elaboration: how step (A) is accomplished, how we check a subset $V_i$ for its category, and how $q$ is chosen. We shall deal with the first two points in § 5.2, and the last point in § 5.3.

**5.2. A set partition theorem.** We shall show that step (A) of the preprocessing phase in § 5.1 can be accomplished. The key is the following result in Yao and Yao [20]. For completeness, a proof is included.

For any finite set $F$ of points in $E^k$, let $\text{high}_l(F) = \max\{x_l \mid \tilde{x} \in F\}$ and $\text{low}_l(F) = \min\{x_l \mid \tilde{x} \in F\}$, for $1 \leqq l \leqq k$.

LEMMA 5.1 [20].[2] *Let $q$ and $k$ be positive integers, and $F$ a set of $n$ points in $E^k$. Then, in $O(kn \log n)$ steps, the following can be done.*

(i) *$F$ is partitioned into $r = \lceil n/q \rceil$ sets $F_1, F_2, \cdots, F_r$, each with at most $q$ points,*

(ii) *The $2kr$ numbers $\text{high}_l(F_i)$, $\text{low}_l(F_i)$, $1 \leqq i \leqq r$, and $1 \leqq l \leqq k$, are computed,*

(iii) *The partition satisfies the condition that, for any $\tilde{y} \in E^k$, there exist at most $k\lceil r^{1/k} \rceil^{k-1}$ sets $F_i$ such that there exists an $l$ with $\text{low}_l(F_i) < y_l \leqq \text{high}_l(F_i)$.*

*Proof.* We shall prove Lemma 5.1 for the case $k = 3$; the extension to general $k$ is obvious. For the moment, let us assume further that $n = qm^3$ for some integer $m$. We use the following procedure to partition $F$.

(a) Sort the points of $F$ in ascending order according to the first components into a sequence $\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_n$. Divide the sorted sequence into $m$ consecutive parts of equal size, that is,

$$G_1 = \{x_j \mid 1 \leqq j \leqq n/m\}, \quad G_2 = \{X_j \mid n/m + 1 \leqq j \leqq 2n/m\}, \quad \cdots, \quad G_m.$$

[2] This lemma was proved in [20] with $q = n^{1/k}$.

(b) For each $1 \leqq i \leqq m$, sort the points in $G_i$ according to the 2nd components; divide the sorted sequence of $G_i$ into $m$ consecutive parts of equal size, $G_{i1}, G_{i2}, \cdots, G_{im}$.

(c) For each $1 \leqq i, j \leqq m$, sort the points in $G_{ij}$ according to their 3rd components; divide the sorted sequence of $G_{ij}$ into $m$ consecutive parts of equal size, $G_{ij1}, G_{ij2}, \cdots, G_{ijm}$.

(d) Rename the $m^3$ sets $G_{ijl}$ as $F_1, F_2, \cdots, F_r$ where $r = n/q = m^3$.

(e) Compute $\text{high}_l(F_i), \text{low}_l(F_i)$ for $1 \leqq i \leqq r, 1 \leqq l \leqq 3$ according to the definitions. The above procedure takes $O(n \log n)$ steps, and each $F_i$ contains exactly $q$ points. It remains to show that property (iii) in Lemma 5.1 is satisfied.

Let $\tilde{y} \in E^3$. We shall prove that, for each $1 \leqq l \leqq 3$, there are at most $m^2$ $F_i$ with $\text{low}_l(F_i) < y_l \leqq \text{high}_l(F_i)$. The proof is based on the following properties of the partition, where $1 \leqq i, j \leqq m$:

$$\text{low}_1(G_1) \leqq \text{high}_1(G_1) \leqq \text{low}_1(G_2) \leqq \text{high}_1(G_2) \leqq \cdots \leqq \text{low}_1(G_m) \leqq \text{high}_1(G_m), \tag{5.1}$$

$$\text{low}_2(G_{i1}) \leqq \text{high}_2(G_{i1}) \leqq \text{low}_2(G_{i2}) \leqq \text{high}_2(G_{i2}) \leqq \cdots \leqq \text{low}_2(G_{im}) \leqq \text{high}_2(G_{im}), \tag{5.2}$$

$$\text{low}_3(G_{ij1}) \leqq \text{high}_3(G_{ij1}) \leqq \text{low}_3(G_{ij2}) \leqq \text{high}_3(G_{ij2}) \leqq \cdots \leqq \text{low}_3(G_{ijm}) \leqq \text{high}_3(G_{ijm}). \tag{5.3}$$

For $l = 1$, according to (5.1), there is at most one $j$ such that

$$\text{low}_1(G_j) < y_1 \leqq \text{high}_1(G_j).$$

Thus, only the $m^2$ $G_{jts}$, $1 \leqq t, s \leqq m$, can have $\text{low}_1(G_{jts}) < y_1 \leqq \text{high}_1(G_{jts})$. This proves our assertion for $l = 1$. We now prove the case for $l = 2$. For each $i$, by (5.2), there is at most one $j$ such that $\text{low}_2(G_{ij}) < y_2 \leqq \text{high}_2(G_{ij})$. Thus, for each $i$, only the $m$ $G_{ijt}$, $1 \leqq t \leqq m$, may have $\text{low}_2(G_{ijt}) < y_2 \leqq \text{high}_2(G_{ijt})$. Therefore, at most $m^2$ $G_{ijt}$ can have $\text{low}_2(G_{ijt}) < y_2 \leqq \text{high}_2(G_{ijt})$. A similar proof works for $l = 3$, making use of formula (5.3).

This proves that, when $k = 3$, and $n = qr = qm^3$ for some integer $m$, Lemma 5.1 is true. We now drop the restriction on $n$ (still $k = 3$). In this situation, $r = \lceil n/q \rceil$. Let $m = \lceil r^{1/k} \rceil$, and use the same procedure. At most $3m^2$ $G_{ijt}$ will satisfy (iii) by the same proof. This completes the proof for $k = 3$. □

We now extend the above result. Let $B = \{\tilde{b}_1, \tilde{b}_2, \cdots, \tilde{b}_k\}$ be a basis of $E^k$; for any $\tilde{x} \in E^k$, we shall define a $k$-tuple $(x'_1, x'_2, \cdots, x'_k)$ by $\tilde{x} = \sum_{i=1}^{k} x'_i \tilde{b}_i$. For any finite set $F$ of points, define for each $1 \leqq l \leqq k$,

$$\text{high}_l(B; F) = \max\{x'_l \mid \tilde{x} \in F\},$$

$$\text{low}_l(B; F) = \min\{x'_l \mid \tilde{x} \in F\}.$$

THEOREM 5.2. *Let $q, n, k$ $(q, k \leqq n)$ be positive integers, $B$ a basis of $E^k$, and $V$ a set of $n$ points in $E^k$. Then, in $O(kn \log n + k^2 n + k^3)$ steps, we can accomplish the following:*

(i) *$V$ is partitioned into $r = \lceil n/q \rceil$ sets $V_1, V_2, \cdots, V_r$, each with at most $q$ points.*

(ii) *The $2kr$ numbers $\text{high}_l(B, V_i), \text{low}_l(B, V_i)$, $1 \leqq i \leqq r, 1 \leqq l \leqq k$, are computed.*

*Furthermore, the partition satisfies the condition*:

(iii) *For any k-tuple of numbers* $(y_1, y_2, \cdots, y_k)$, *there exist at most* $k\lceil r^{1/k}\rceil^{k-1}$ $V_i$ *such that there exists an* $l$ *that satisfies*:

$$\text{low}_l(B; V_i) < y_l \leqq \text{high}_l(B; V_i).$$

Before proving this theorem, let us check that this partition fulfills the requirements of step (i) in the preprocessing phase (see § 5.1).

LEMMA 5.3. *A point* $\tilde{y}$ *is in the region* $B$ *to* $\tilde{x}$, *i.e.*, $\tilde{y} \in R(B; \tilde{x})$, *if and only if* $y'_l \geqq x'_l$ *for all* $1 \leqq l \leqq k$.

*Proof.* The lemma follows from the equation $\tilde{y} - \tilde{x} = \sum_{l=1}^{k} (y'_l - x'_l)\tilde{b}_l$. □

LEMMA 5.4. *If* $\tilde{x} \in E^k$, $B$ *a basis, and* $F$ *a finite set of points in* $E^k$, *then either*

(i) $x'_l \leqq \text{low}_l(B; F)$ *for all* $1 \leqq l \leqq k$, *in which case all points in* $F$ *are in region* $B$ *to* $\tilde{x}$, *or*

(ii) $\exists l$, $x'_l > \text{high}_l(B; F)$, *in which case none of the points in* $F$ *is in region* $B$ *to* $\tilde{x}$, *or*

(iii) *none of the above*; *there exists an* $l$ *such that* $\text{low}_l(A; F) < x'_l \leqq \text{high}_l(B; F)$.

*Proof.* This is an immediate consequence of Lemma 5.3. □

There are two consequences of Lemma 5.4 of interest to us. Firstly, the lemma shows that the requirements of step (A) in § 5.1 are satisfied. For any $\tilde{x}$, a $V_j$ such that *neither* all points of $V_j$ are in $R(B; \tilde{x})$ *nor* none are in $R(B; \tilde{x})$ must satisfy the condition that $\text{low}_l(B, V_j) < x'_l \leqq \text{high}_l(B; V_j)$ for some $l$, because of Lemma 5.4. By Theorem 5.2, there are at most about $r^{1-1/k}$ such $V_j$. This proves the claim. Secondly, Lemma 5.4 gives a simple way to detect most of the $V_j$ that satisfy $V_j \subseteq R(B; \tilde{x})$ or $V_j \cap R(B; \tilde{x}) = \varnothing$. Namely, compare $x'_l$ with $\text{high}_l(B; V_j)$ and $\text{low}_l(B; V_j)$ for all $l$, and determine whether case (i), (ii), or (iii) applies in Lemma 5.4. The test only takes $O(k)$ for each $i$ and $j$, can be conveniently used in step (C) in the procedure in § 5.1.

We now turn to the proof of Theorem 5.2.

*Proof of Theorem* 5.2. Let $M$ be the $k \times k$ matrix $(b_{ij})$ (recall that $\tilde{b}_i = (b_{i1}, b_{i2}, \cdots, b_{ik})$), and $M^{-1}$ be its inverse. We use the following procedure to partition $V$.

(1) Compute $M^{-1}$ in $O(k^3)$ steps (see, e.g., [1]).

(2) Compute, for each $\tilde{x} \in V$, the $k$-tuple $(x'_1, x'_2, \cdots, x'_k)$ by $(x'_1, x'_2, \cdots, x'_k) = (x_1, x_2, \cdots, x_k) \cdot M^{-1}$. This takes $O(k^2 n)$ steps.

(3) Consider the set $F = \{(x'_1, x'_2, \cdots, x'_k) \mid \tilde{x} \in V\}$. We now use the procedure in Lemma 5.1 to divide $F$ into $r$ parts $F_1, F_2, \cdots, F_r$. Let $V_i$ be the subset of $V$ obtained from $F_i$ by replacing every $(x'_1, \cdots, x'_k)$ by the corresponding $\tilde{x}$.

(4) Set $\text{high}_l(B; V_i) \leftarrow \text{high}_l(F_i)$, and $\text{low}_l(B; V_i) \leftarrow \text{low}_l(F_i)$.

The procedure clearly takes $O(kn \log n + k^2 n + k^3)$ steps. The quantities $\text{high}_l(B; V_i)$ and $\text{low}_l(B; V_i)$ are correctly computed by their definitions. Items (i) and (ii) in Theorem 5.2 are obviously true, and (iii) is true because of the properties of $\text{high}_l(F_i)$, $\text{low}_l(F_i)$ stated in Lemma 5.1. □

**5.3. Finishing the proof.** We now analyze the running time of the algorithm for fixed $k$ and choose $q$. The preprocessing phase takes time $O(n \log n + r \cdot q^{b(k)})$. In the finishing phase, the running time is dominated by the search for candidates $\tilde{w}$, which is of order $n[(\#\text{ of category-1 } V_j) \cdot \log q + (\#\text{ of category-3 } V_j) \cdot q]$. The last expression is bounded by $n(r \log q + r^{1-k^{-1}} \cdot q)$. The total running time of the algorithm is thus $O(n \log n + r \cdot q^{b(k)} + nr \log q + nqr^{1-k^{-1}})$. Remembering that $b(k) = 2^{k+1}$ and $r = O(n/q)$, we optimize the expression by choosing $q \approx (n \log n)^{a(k)}$. This gives a time $O(n^{2-a(k)} (\log n)^{1-a(k)})$. The improved time bound for the special case $k = 3$, $p = 2$ can be similarly obtained.

**6. Discussions.** We have shown that, for fixed $k$ and $p \in \{1, 2, \infty\}$, there are $o(n^2)$-time algorithms for a number of geometric problems in $E_p^k$, including the minimum spanning tree problem. We shall now argue that, when $p \in \{2, \infty\}$, $o(kn^2)$ algorithms exist for all $k$ and $n$. As are typical for results under fixed $k$ assumptions, the algorithms given in the paper have $o(n^2)$ time bounds when $k$ is allowed to grow slowly with $n$. In fact, a close examination shows that, if $k \leqq \frac{1}{2} \log \log n$, the algorithms still run in time $o(n^2)$. For $k > \frac{1}{2} \log \log n$, it can be shown [18] that the computation of the distances between all points can be done in $o(kn^2)$ time when $p \in \{2, \infty\}$. Since all problems considered in this paper have $O(n^2)$ algorithms once all the distances are known, the previous statement provides algorithms that run in time $o(kn^2)$.

The efficiency of our algorithms is dependent on the solution to the post-office problem (or its farthest-point analogue). For example, suppose the nearest-point query could be answered in $O(\log n)$ time after an $O(n^\beta)$-time preprocessing, $\beta \geqq 2$. A simple adaptation of the algorithm would give an $O(n^{2-\beta^{-1}}(\log n)^{1-\beta^{-1}})$-time solution to the NFN-problem, which in turn implies an $O((n \log n)^{2-\beta^{-1}})$-time solution to the MST-problem (see the remark at the end of § 2). If $1 < \beta < 2$, the following modification would also give an $O(n^{2-\beta^{-1}}(\log n)^{1-\beta^{-1}})$-algorithm for the NFN-problem (and hence an $O((n \log n)^{2-\beta^{-1}})$-algorithm for finding MST). We first divide $V$ into $r \approx n/(n \log n)^{\beta^{-1}}$ blocks $B_1, B_2, \cdots$ as before. Each block is preprocessed, and for each $\tilde{x}$, a nearest point in every block not containing $x$ is found. Now, for every point $\tilde{x} \in B_i$, we need to find for it a nearest "foreign" neighbor in $B_i$. Instead of using brute force (computing the distance from each $\tilde{x} \in B_i$ to every other point in $B_i$) as was done previously, we divide $B_i$ into $r$ subblocks, preprocess each subblock, and find for $\tilde{x}$ a nearest point in every subblock in $B_i$. To compute a nearest foreign neighbor to $\tilde{x}$ in the subblock containing $\tilde{x}$, we shall again break the subblocks. This process continues until the size of the subblocks is less than $n^\delta$, where $\delta = 1 - \beta^{-1}$, at which point we compute all distances between points in the same subblock. During the above process, we have located, for each $\tilde{x}$, a set of points containing a nearest foreign neighbor $\tilde{u}$ to $\tilde{x}$. It is then simple to locate such a $\tilde{u}$. This is a brief outline of an $O(n^{2-\beta^{-1}}(\log n)^{1-\beta^{-1}})$-algorithm for NFN-problems, $1 < \beta < 2$. However, it seems unlikely that a nearest-point query can be answered in $O(\log n)$ time with an $O(n^\beta)$-preprocessing, $\beta < 2$, when $k \geqq 3$.

We conclude this paper with the following open problems.

(1) Improve the bounds obtained in this paper.

(2) Analyze the performance of new or existing fast heuristic algorithms for MST-problems. For example, can one show that the AMST-algorithm in [2] always constructs a spanning tree with length at most 5% over the true MST?

(3) Prove bounds on average running time of MST-algorithms for some natural distributions.

(4) Extend results in this paper to $L_p$-metric for general $p$.

## Appendix. The existence and construction of "narrow" frames—Proof of Lemma 4.2.

LEMMA 4.2. *For any $0 < \psi < \pi$, one can construct in finite steps a frame $\mathcal{B}$ of $E_p^k$ such that* $\text{Ang}(\mathcal{B}) < \psi$.

As the discussion is independent of $p$, we shall use $E^k$ instead of $E_p^k$.

We begin with the concept of a "simplex" familiar in Topology (see, e.g., [10]). Let $\tilde{p}_0, \tilde{p}_1, \cdots, \tilde{p}_j$ be $j+1$, $0 \leqq j \leqq k$, points in $E^k$, where the vectors $\tilde{p}_i - \tilde{p}_0$, $1 \leqq i \leqq j$, are linearly independent. We shall call the set $\{\sum_{i=0}^{j} \lambda_i p_i \mid \lambda_i \geqq 0 \text{ for all } i, \text{ and } \sum_i \lambda_i = 1\}$ a (geometric) *$j$-simplex in $E^k$*, denoted by $\langle \tilde{p}_0, \tilde{p}_1, \cdots, \tilde{p}_j \rangle$. Informally, it is the convex

hull formed by vertices $\tilde{p}_0, \tilde{p}_1, \cdots, \tilde{p}_j$ on the minimal linear subspace containing them (see Fig. A). The *diameter* of a simplex $s$ is $\text{diam}(s) = \sup\{\|\tilde{x} - \tilde{y}\| \mid \tilde{x}, \tilde{y} \in s\}$.

The following two lemmas give the connection between simplices and bases. Let $\hat{\varepsilon}$ be a $k$-tuple $(\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_k)$, where $\varepsilon_i \in \{-1, 1\}$ for all $i$. Denote by $H(\hat{\varepsilon})$ the hyperplane $\{x \mid \sum_i \varepsilon_i x_i = 1\}$ in $E^k$.

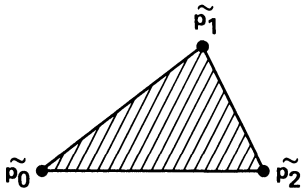


FIG A. *A 2-simplex in* $E^2$

LEMMA A1. *Let* $s = \langle \hat{p}_0, \hat{p}_1, \cdots, \hat{p}_{k-1} \rangle$ *be a* $(k-1)$*-simplex in* $E^k$, *where* $\tilde{p}_i \in H(\hat{\varepsilon})$ *for every* $i$. *Then the set* $B(s) = \{\tilde{p}_0, \tilde{p}_1, \cdots, \tilde{p}_k\}$ *is a basis. Furthermore, the angle* $\varphi = \text{Ang}(B(s))$ *satisfies* $\cos\varphi \geqq 1 - \frac{1}{2}k(\text{diam}(s))^2$.

*Proof.* Suppose $\sum_{i=0}^{k-1} \lambda_i \tilde{p}_i = 0$. We shall show that $\lambda_i = 0$ for all $i$. If $\sum_{i=0}^{k-1} \lambda_i = 0$, then $\sum_{i=1}^{k-1} \lambda_i(\tilde{p}_i - \tilde{p}_0) = \sum_{i=0}^{k-1} \lambda_i \tilde{p}_i = 0$. This implies $\lambda_i = 0$ for all $i$, by the definition of simplex. If $\sum_{i=0}^{k-1} \lambda_i = \Lambda \neq 0$, then $\tilde{v} = \sum_{i=0}^{k-1} (\lambda_i/\Lambda)\tilde{p}_i = 0$. But it is easy to check that $v \in H(\hat{\varepsilon})$, a contradiction.

We have thus shown that $B(s)$ is a basis. To prove the rest of the lemma, let $\tilde{x}$ and $\tilde{y}$ be any two nonzero vectors in $\text{Conv}(B(s))$. We shall prove that $\cos\theta(\tilde{x}, \tilde{y}) \geqq 1 - \frac{1}{2}k(\text{diam}(s))^2$. Without loss of generality, we can assume that $\tilde{x}, \tilde{y} \in s$. Then

$$\begin{aligned}(\text{diam}(s))^2 &\geqq (\tilde{x} - \tilde{y}) \cdot (\tilde{x} - \tilde{y}) = \|\tilde{x}\|^2 + \|\tilde{y}\|^2 - 2\|\tilde{x}\| \cdot \|\tilde{y}\| \cos\theta(\tilde{x}, \tilde{y}) \\ &\geqq 2\|\tilde{x}\| \cdot \|\tilde{y}\|(1 - \cos\theta(\tilde{x}, \tilde{y})).\end{aligned}$$

It follows that

$$\cos\theta(\tilde{x}, \tilde{y}) \geqq 1 - \frac{(\text{diam}(s))^2}{2\|\tilde{x}\| \cdot \|\tilde{y}\|}. \tag{A1}$$

As can be easily verified, $\tilde{x}, \tilde{y} \in H(\hat{\varepsilon})$, which implies

$$\|\tilde{x}\|^2 = \sum_i x_i^2 \geqq \frac{1}{k}\left(\sum_i \varepsilon_i x_i\right)^2 = \frac{1}{k}.$$

Therefore, $\|\tilde{x}\| \geqq 1/\sqrt{k}$, and similarly $\|\tilde{y}\| \geqq 1/\sqrt{k}$. Formula (A1) then implies

$$\cos\theta(\tilde{x}, \tilde{y}) \geqq 1 - \frac{k}{2}(\text{diam}(s))^2.$$

This proves Lemma A1. □

We shall use $B(s)$ to denote the basis corresponding to simplex $s$.

LEMMA A2. *Let* $s \subseteq H(\hat{\varepsilon})$ *be a simplex,* $\mathcal{S}$ *a finite collection of simplices, and* $s = \bigcup_{s' \in \mathcal{S}} s'$. *Then* $\text{Conv}(B(s)) = \bigcup_{s' \in \mathcal{S}} \text{Conv}(B(s'))$.

*Proof.* It is easy to see that $\text{Conv}(B(s)) \supseteq \bigcup_{s' \in \mathcal{S}} \text{Conv}(B(s'))$. To prove the converse, let $s = \langle \tilde{p}_0, \tilde{p}_1, \cdots, \tilde{p}_{k-1} \rangle$, where each $\tilde{p}_i \in H(\hat{\varepsilon})$. If a point $\tilde{u} \in \text{Conv}(B(s))$, then $\tilde{u} = \sum_{i=0}^{k-1} \lambda_i \tilde{p}_i$, where $\lambda_i \geqq 0$. We shall prove that $\tilde{u} \in \text{Conv}(B(s'))$ for some $s' \in \mathcal{S}$. It is trivial if $\tilde{u} = 0$. Otherwise, the point $(1/\sum_i \lambda_i)\tilde{u} \in s = \bigcup_{s' \in \mathcal{S}} s'$, and hence $(1/\sum_i \lambda_i)\tilde{u} \in s'$ for some $s' \in \mathcal{S}$. This implies $\tilde{u} \in \text{Conv}(B(s'))$. □

The above lemmas suggest that we may try to construct a frame with narrow bases, by first constructing a family of simplices all with small diameters. We use the following scheme:

Let $\tilde{e}_i$ denote the unit vector in $E^k$, whose $i$th component is 1 and all others are 0.

For each of the $2^k$ $k$-tuples $\hat{\varepsilon} = (\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_k)$, where $\varepsilon_i = \pm 1$, do the following.

(a) Let $s = \langle \varepsilon_1 \tilde{e}_1, \varepsilon_2 \tilde{e}_2, \cdots, \varepsilon_k \tilde{e}_k \rangle$. (Clearly $s \subseteq H(\hat{\varepsilon})$.)

(b) Construct a finite family $\mathcal{S}$ of simplices all contained in $H(\hat{\varepsilon})$ such that $s = \bigcup_{s' \in \mathcal{S}} s'$ and diam $(s') < (2(1-\cos\psi)/k)^{1/2}$ for all $s' \in \mathcal{S}$.

(c) Form $B(s')$ for all $s' \in \mathcal{S}$.

The collection $\mathcal{B}$ of all the $B(s')$ constructed this way is clearly a frame because of Lemma A2. Using Lemma A1, it is easy to verify that Ang $(B(s')) < \psi$ for all $s'$. Thus, such a construction would give a frame satisfying the conditions in Lemma 4.2. It remains to show that step (b) above can be carried out.

A procedure in topology ([10, p.209, Thm. 5–20]), known as *barycentric subdivision*, guarantees that step (b) can be accomplished in a finite number of steps. For completeness, we shall give a brief description below.

There is a basis procedure, called *first barycentric subdivision* (FBS), which, for a given $j$-simplex $s$, constructs in finite steps a family $\mathcal{S}$ of simplices such that $s = \bigcup_{s' \in \mathcal{S}} s'$ and $\max_{s' \in \mathcal{S}} (\text{diam } (s')) \leqq (j/(j+1))(\text{diam } (s))$. If we apply this procedure iteratively, at each iteration we apply FBS to every simplex present, then all the simplices will have a diameter less than any prescribed positive number after enough iterations. This then constitutes a procedure for step (b).

Finally, we describe the FBS procedure. For a proof that it produces simplices with the desired properties, see [10]. Let $s = \langle \tilde{p}_0, \tilde{p}_1, \cdots, \tilde{p}_j \rangle$, the point $\tilde{c}(s) = (1/(j+1)) \sum_{i=0}^{j} \tilde{p}_i$ is called the *centroid* of simplex $s$. For any $t$ distinct integers $0 \leqq i_1, i_2, \cdots, i_t \leqq j$, let $\tilde{p}_{i_1 i_2 \cdots i_t} = \tilde{c}(\langle \tilde{p}_{i_1}, \tilde{p}_{i_2}, \cdots, \tilde{p}_{i_t} \rangle)$. For each $\sigma = (i_0, i_1, \cdots, i_j) \in \Sigma$, where $\Sigma$ is the set of all permutations of $(0, 1, 2, \cdots, j)$, let $s'(\sigma)$ denote the simplex $\langle \tilde{p}_{\sigma_0}, \tilde{p}_{\sigma_1}, \cdots, \tilde{p}_{\sigma_j} \rangle$ with $\sigma_t = i_0, i_1, \cdots, i_t$. The FBS of $s$ is defined by

$$\mathcal{S} = \{ s'(\sigma) \mid \sigma \in \Sigma \}.$$

## REFERENCES


[1] A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] J. L. BENTLEY AND J. H. FRIEDMAN, *Fast algorithms for constructing minimal spanning trees in coordinate spaces*, Rep. STAN-CS-75-529, Stanford Computer Science Department, Stanford, CA, January 1976.

[3] J. L. BENTLEY AND M. I. SHAMOS, *Divide-and-conquer in multidimensional space*, in Proc. 8th Annual ACM Symposium on Theory of Computing, 1976, pp. 220–230.

[4] C. BERGE AND A. GHOUILA-HOURI, *Programming, Games, and Transportation Networks*, John Wiley, New York, 1965.

[5] R. C. BUCK, *Partition of space*, Amer. Math. Monthly, 50 (1943), pp. 541–544.

[6] D. CHERITON AND R. E. TARJAN, *Finding minimum spanning trees*, this Journal, 5 (1976), pp. 724–742.

[7] E. W. DIJKSTRA, *A note on two problems in connection with graphs*, Numer. Math., 1 (1959), pp. 269–271.

[8] D. DOBKIN AND R. J. LIPTON, *Multidimensional search problems*, this Journal, 5 (1976), pp. 181–186.

[9] R. O. DUDE AND P. E. HART, *Pattern Classification and Science Analysis*, John Wiley, New York, 1973.

[10] J. G. HOCKING AND G. S. YOUNG, *Topology*, Addison-Wesley, Reading, MA, 1961.

[11] A. KERSCHENBAUM AND R. VAN SLYKE, *Computing minimum spanning trees efficiently*, in Proc. 25th Annual Conference of the ACM, 1972, pp. 518–527.

[12] D. E. KNUTH, *The Art of Computer Programming, Vol. 3, Sorting and Searching*, Addison-Wesley, Reading, MA, 1973.
[13] J. B. KRUSKAL, JR7., *On the shortest spanning subtree of a graph and the travelling salesman problem*, Problem. Amer. Math. Soc., 7 (1956), pp. 48–50.
[14] R. C. PRIM, *Shortest connection networks and some generalizations*, Bell System Tech. J., 36 (1957), pp. 1389–1401.
[15] M. I. SHAMOS, *Geometric complexity*, Proc. 7th Annual ACM Symposium on Theory of Computing, 1975, pp. 224–233.
[16] M. I. SHAMOS AND D. J. HOEY, *Closest-point problems*, Proc. 16th Annual IEEE Symposium on Foundations of Computer Science, 1975, pp. 151–162.
[17] A. C. YAO, *An $O(|E| \log\log |V|)$ algorithm for finding minimum spanning trees*, Inform. Process. Lett., 4 (1975); pp. 21–23.
[18] ———, *On computing the distance matrix of n points in k dimensions*, IBM San Jose Research Center Technical Report, to appear.
[19] ———, *On the preprocessing cost in multidimensional search*, IBM San Jose Research Center Technical Report, to appear.
[20] A. C. YAO AND F. F. YAO, *On computing the rank function for a set of vectors* Rep. UIUCDCS-R-75-699, Computer Science Department, University of Illinois, Illinois, Februrary, 1975.
[21] C. T. ZAHN, *Graph-theoretical method for detecting and describing gestalt clusters*, IEEE Trans. Comput., C-20 (1970), pp. 68–86.