SIAM J. COMPUT.
Vol. 17, No. 3, June 1988

© 1988 Society for Industrial and Applied Mathematics
010

Downloaded 03/28/24 to 24.18.43.38 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

MINIMUM SPANNING TREES IN k-DIMENSIONAL SPACE*

PRAVIN M. VAIDYA†

Abstract. We study the problem of finding a minimum spanning tree in the complete graph on a set V of n points in k-dimensional space. The points are the vertices of this graph and the weight of an edge between two points is the distance between the points under some L_q metric. We give an $O(\varepsilon^{-k} n \log n)$ algorithm for finding an approximate minimum spanning tree in such a graph; the weight of the approximate minimum spanning tree is guaranteed to be at most $(1+\varepsilon)$ times the weight of a minimum spanning tree. We also present an algorithm to find a minimum spanning tree in the complete graph on V. Under the assumption that V consists of n random points, independently and uniformly distributed in the unit k-cube $[0, 1]^k$, the expected running time of this minimum spanning tree algorithm is shown to be $O(n\alpha(cn, n))$ where c is a constant dependent on k and α is the inverse Ackermann function.

Key words. minimum spanning trees, approximation algorithms

AMS(MOS) subject classifications. 68Q20, 68Q25, 68U05

1. Introduction.

Given an undirected graph with a weight assigned to each edge, a spanning tree is a connected acyclic subgraph, and 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 an MST were given by Dijkstra [4], Kruskal [9], Prim [10], and Sollin [1]. It is well known that for a graph on n vertices, an MST may be found in $O(n^2)$ time. For a graph with m edges and n vertices, it was shown by Yao [15] that an MST may be found in $O(m \log \log n)$ time. Further results on MST's may be found in [6], [8].

We study the problem of finding an MST in the complete graph on a given set V of n points in k-dimensional space. The points are the vertices of this graph and the weight of an edge between any two points is the distance between the points under some distance metric L_q. Each point x is given as a vector $(x_1, x_2, \cdots, x_k)$. The L_q, $q = 1, 2, \cdots, \infty$, distance between any two points x and y is given by $(\sum_{i=1}^{k} |x_i - y_i|^q)^{1/q}$. (Note that the L_∞ distance is given by $\max_i |x_i - y_i|$.) We assume that the dimension k and the distance metric L_q are fixed (so distance is to be always interpeted as L_q distance).

The problem of finding an MST on a set of points in k-dimensional space differs from the problem of finding an MST in a general weighted undirected graph in two respects. First, the input consists only of kn numbers, the edges and the edge weights being implicitly defined. Second, in many applications of MST on points in space, like clustering, pattern recognition [5], [17], and other geometric and statistical applications, a spanning tree whose weight is close to the weight of an MST would serve just as well. So it is useful and interesting to investigate if the geometric nature of the problem can be exploited to obtain fast algorithms for finding a spanning tree on points in space whose weight is minimum or close to minimum.

Shamos and Hoey were the first to utilize the geometric nature of this problem, and in [12] they give an $O(n \log n)$ algorithm for n points in the plane ($k = 2$) with Euclidian metric. In [16] Yao gives algorithms which construct an MST in time

* Received by the editors August 18, 1986; accepted for publication April 17, 1987. The work of this author was supported by a fellowship from the Shell Foundation.

† Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801. Present address, AT&T Bell Laboratories, Murray Hill, New Jersey 07974.

$O(n^{2-2^{-(k+1)}}(\log n)^{1-2^{-(k+1)}})$ for any fixed $k \geqq 3$, and distance metrics L_q, $q = 1, 2, \infty$. In [7], $O(n(\log n)^{k-\theta})$, $\theta \leqq 2$, algorithms are given for fixed $k \geqq 2$ and L_1, L_∞ distance metrics. An algorithm for finding a spanning tree, with weight at most $(1+\varepsilon)$ times minimum, for $k=3$ and L_2 distance metric is given in [2] but the running time is dependent on the ratio of the maximum to the minimum distance between any two points. In [14], an $O(n(\log n)^{k+1}\varepsilon^{-(k-1)})$ algorithm is given for finding a spanning tree with weight at most $(1+\varepsilon)$ times minimum, for $\varepsilon > 0$, and fixed k and fixed metric L_q.

We give two algorithms for fixed dimension k and fixed metric L_q. The first algorithm, given in § 3, runs in $O(\varepsilon^{-k} n \log n)$ time and finds an approximate minimum spanning tree whose weight is at most $(1+\varepsilon)$ times the weight of an MST. The second algorithm, presented in § 4.1, always finds an MST. For n random points, independently and uniformly distributed in the unit k-cube $[0, 1]^k$, the expected running time of the second algorithm is shown to be $O(n\alpha(cn, n))$, where c is a constant dependent on k and α is the inverse Ackermann function defined in [13]. As α grows extremely slowly with n, the expected running time of the second algorithm is almost linear. We note that the constants in the running times of both the algorithms depend on the dimension k. The probabilistic analysis of the second algorithm is given in § 4.2.

As far as the model of computation is concerned, we assume that all access, arithmetic and comparison operations require constant time. We also assume that some form of indirect addressing is available so that the process of distributing n numbers into m buckets can be carried out in $O(m+n)$ time.

Without loss of generality we assume that all the n points in the given set V are located in the unit k-cube $[0, 1]^k$. We let $d(p, p')$ denote the distance between two points p and p', and we let W_{MST} denote the weight (length) of an MST in the complete graph on V. A box is defined to be the product $J_1 \times J_2 \times \cdots \times J_k$ of k intervals, or alternatively the set of those points $x = (x_1, x_2, \cdots, x_k)$ such that x_i is in interval J_i, $i = 1, 2, \cdots, k$. A box is cubical if and only if all the k intervals defining it have the same length, and the size of a cubical box is the length of each of the k intervals defining it. For a set of points S, we let $d_{\max}(S)$ denote the greatest distance between two points in S. For sets of points S_1 and S_2, we let $d_{\min}(S_1, S_2)$ and $d_{\max}(S_1, S_2)$, respectively, denote the minimum and maximum distance between a point in S_1 and a point in S_2.

2. A brief overview.

In the approximate minimum spanning tree algorithm we first extract a sparse graph $G = (V, E)$ from the given set of points V, and then find an MST in G using a standard procedure [6], [7], [11]. There are $O(\varepsilon^{-k} n)$ edges in G and there is a spanning tree in G of weight at most $(1+\varepsilon) W_{\mathrm{MST}}$. We obtain the graph G as follows. Using a collection of grids the region containing the given points is divided into cubical boxes, each grid partitioning the region into boxes of identical size. In each box b, we select a representative point from among the points in V that are located in box b. An edge between the representative in box b and the representative in box b' is included in G if and only if b and b' are of identical dimensions and the minimum distance between b and b' is below a certain threshold. In addition to edges between representatives, G contains an adequate number of short edges (of length $< \varepsilon W_{\mathrm{MST}}/3n^2$) suitably chosen to ensure that G is connected. Now suppose V is partitioned into V_1 and V_2, and (p_1, p_2), $p_1 \in V_1$, $p_2 \in V_2$, is the unique edge in an MST from V_1 to V_2. Then either there are points $p_1' \in V_1$, $p_2' \in V_2$, such that (p_1', p_2') is in G, p_1' and p_2' are representative points in boxes, and $d(p_1', p_2') \leqq (1+\varepsilon) d(p_1, p_2)$, or among the short edges in G there is a path from p_1 to p_2 of length at most $\varepsilon W_{\mathrm{MST}}/n^2$. This guarantees the existence of a good spanning tree in G.

Downloaded 03/28/24 to 24.18.43.38 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

The procedure to extract G may be implemented in $O(\varepsilon^{-k}n \log n)$ time, and as G contains $O(\varepsilon^{-k}n)$ edges, an MST in G may be found in $O(\varepsilon^{-k}n \log n)$ time using a standard procedure [6], [8], [11], [15]. So the overall running time of the approximate minimum spanning tree algorithm is $O(\varepsilon^{-k}n \log n)$.

In the exact minimum spanning tree algorithm we first extract a graph $G' = (V, E')$ from the given set of points V such that G' always contains a spanning tree of weight W_{MST}. The graph G' is sparse with high probability, and can be extracted in $O(|E'|)$ time. We then try to sort the edges in G' by weight, in $O(|E'|)$ (linear) time, by running a fixed number of passes of radix (bucket) sort [11], with radix $2^{\lceil \log_2 n \rceil}$. If this approach fails to sort the edges in G', we sort them in $O(|E'|\log|E'|)$ time using a standard algorithm [11]. Once a sorted list of edges in G' is available, utilizing Kruskal's algorithm an MST in G' may be obtained in $O(n\alpha(|E'|, n))$ time where α is the inverse Ackermann function defined in [13]. Suppose the given set V consists of n random points, independently and uniformly distributed in the unit k-cube $[0, 1]^k$. Then the probability that G' has more than cn edges, where c is a constant dependent on the dimension k, is $o(1/n^2)$. Also, the probability that a fixed number of passes of radix sort, with radix $2^{\lceil \log_2 n \rceil}$, fail to sort the edges is $o(1/n^2)$. Then it follows that the expected running time of the minimum spanning tree algorithm is $O(n\alpha(cn, n))$.

G' is obtained in a manner similar to G above. Using a collection of grids the unit k-cube $[0, 1]^k$ is divided into cubical boxes, each grid partitioning the unit k-cube into boxes of identical size. Let b_1, b_2, be boxes of identical dimensions such that the minimum distance between b_1 and b_2 is above a certain threshold and below another threshold. We test an easy to compute condition such that (i) if the condition is false, then none of the edges between a point in $b_1 \cap V$ and a point in $b_2 \cap V$ can be included in an MST in the complete graph on V, whereas (ii) if the condition is true, then there is an empty region which does not contain a point in V and whose volume is greater than or equal to the volume of b_1 or b_2. We include every edge between a point in $b_1 \cap V$ and a point in $b_2 \cap V$ in G' if and only if this condition holds. In addition, we also include most of the short edges (length $< c'n^{-1/k}$, c' a constant) in G'. We thereby ensure that G' always contains an MST in the complete graph on V. If the number of edges in G' exceeds cn, then either there is a large region in the unit k-cube which does not contain a point in V or there is some region in the unit k-cube that contains a concentration of points in V, and under the assumption that the points in V are independently and uniformly distributed in the unit k-cube both these events occur with very low probability.

3. Approximate minimum spanning tree algorithm.

The algorithm to find an approximate MST in the complete graph on the given set V of n points consists of two stages: in the first stage we extract a sparse graph $G = (V, E)$ from the given set V of points, and in the second stage we use a standard procedure [6], [7], [11] to find a minimum spanning tree in the graph G. The graph $G = (V, E)$ has the following properties.

(1) G contains a spanning tree whose weight (length) is at most $(1+\varepsilon+1/n)$ times W_{MST}.

(2) $|E| = O(\varepsilon^{-k}n)$.

We shall require a few definitions before we can describe the algorithm to extract G. Let g_0 be a smallest cube enclosing all the n points in V, and let L_0 be the length of a side of g_0. Let g_i be a grid that partitions g_0 into 2^{ki} identical cubical boxes, and let $\delta = \lceil (\log_2 (24(\lceil \varepsilon^{-1} \rceil)^2 k^2 n^2)) \rceil$. Let B_i denote the set of those boxes (cubes) in g_i which contain a point in V, and let $B = \bigcup_{i=0}^{\delta} B_i$. Let $L_i = c_a \lceil \varepsilon^{-1} \rceil 2^{-i} L_0$, where $c_a = 6k^{1/q}$

Downloaded 03/28/24 to 24.18.43.38 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

for L_q metric, $q = 1, 2, 3, \cdots, \infty$. We let $r(b)$ denote the representative point in box b. We now give an algorithm to extract the graph $G = (V, E)$ from the set of points V.

```
ALGORITHM SPARSE-GRAPH.
1. For each box $b \in B$, pick a representative point $r(b)$ from among the points
   in $b \cap V$, and let $R_i = \{r(b): b \in B_i\}$, for $i = 1, \cdots, \delta$. We pick representatives
   so that $R_1 \subseteq R_2 \subseteq \cdots \subseteq R_i \subseteq \cdots \subseteq R_\delta$.
2. Let $X = \bigcup_{i=1}^{\delta} X_i$ where
       $X_i = \{(r(b_1), r(b_2)): b_1 \in B_i, b_2 \in B_i, d_{\min}(b_1, b_2) \leqq L_i\}$.
3. Let $Y = \{(p, r(b)): p \in b, b \in B_\delta\}$.
4. Let $E = X \cup Y$.
end Sparse-Graph
```

The edges in X_i connect representatives in boxes b_1, b_2, in B_i such that $d_{\min}(b_1, b_2)$ lies between two thresholds L_i and $L_i/3$. The edges in Y are small in length compared to W_{MST} and they ensure that $G = (V, E)$ is connected.

We need some additional definitions before we can show that G has the desired properties. Let Z denote the set of all the edges in the complete graph on V, and let

$$Z_i = \{(p_1, p_2): p_1 \in b_1, p_2 \in b_2, b_1 \in B_i, b_2 \in B_i, d_{\min}(b_1, b_2) \geqq L_i/3\}.$$

The following two lemmas follow directly from the definitions.

LEMMA 1. *For* $1 \leqq i \leqq \delta$, *if* $b_1 \in B_i$, $b_2 \in B_i$, *and* $d_{\min}(b_1, b_2) \geqq L_i/3$ *then* $d_{\max}(b_1) = d_{\max}(b_2) \leqq (\varepsilon/2) d_{\min}(b_1, b_2)$ and $d_{\max}(b_1, b_2) \leqq (1+\varepsilon) d_{\min}(b_1, b_2)$.

LEMMA 2. $Z_0 \subseteq Z_1 \subseteq \cdots \subseteq Z_i \subseteq \cdots \subseteq Z_\delta \subseteq Z$.

LEMMA 3. *For* $1 \leqq i \leqq \delta$, *if* $(p_1, p_2) \in (Z_i - Z_{i-1})$ *then* p_1, p_2, *are located in boxes* b_1, b_2, *which satisfy* $b_1 \in B_i$, $b_2 \in B_i$, *and* $L_i/3 \leqq d_{\min}(b_1, b_2) \leqq L_i$.

Proof. Suppose p_1, p_2 are located in boxes b_1, b_2, in B_i, respectively. Since $(p_1, p_2) \in Z_i$, $d_{\min}(b_1, b_2) \geqq L_i/3$. Let $b'_1 \in B_{i-1}$, $b'_2 \in B_{i-1}$, and let $b_1 \subseteq b'_1$, $b_2 \subseteq b'_2$. Since $(p_1, p_2) \in (Z_i - Z_{i-1})$, we have $d_{\min}(b'_1, b'_2) \leqq L_{i-1}/3$. Then

$$\begin{aligned} d_{\min}(b_1, b_2) &\leqq d_{\min}(b'_1, b'_2) + d_{\max}(b'_1)/2 + d_{\max}(b'_2)/2 \\ &\leqq L_{i-1}/3 + k^{1/q} 2^{-(i-1)} L_0 \\ &\leqq L_i. \end{aligned}$$ □

We now show that $G = (V, E)$ contains a spanning tree whose weight is at most $(1 + \varepsilon + 1/n) W_{\text{MST}}$. Let T be an MST in the complete graph on V. We shall give a function $f\colon T \to 2^E$ such that the graph $(V, \bigcup_{e \in T} f(e))$ is connected and the sum of the weights of the edges in $\bigcup_{e \in T} f(e)$ is at most $(1 + \varepsilon + 1/n) W_{\text{MST}}$. The function $f(e)$ is defined as follows.

(1) If $e \in T$ and $e \in E$, then we let $f(e) = \{e\}$.

(2) Suppose $e \in ((T - E) \cap (Z - Z_\delta))$. Let $e = (p_1, p_2)$, and let b_1, b_2 be the boxes in B_δ which contain p_1, p_2, respectively. Then $f(e) = \{(p_1, r(b_1)), (r(b_1), r(b_2)), (r(b_2), p_2)\}$ and $f(e) \subseteq (Y \cup X_\delta)$. The length of an edge in Y is at most $\varepsilon W_{\text{MST}}/3n^2$ and $d(r(b_1), r(b_2)) \leqq \varepsilon W_{\text{MST}}/3n^2$, so the sum of the lengths of the edges in $f(e)$ is at most $\varepsilon W_{\text{MST}}/n^2$.

(3) Let $e \in ((T - E) \cap (Z_i - Z_{i-1}))$, $i \leqq \delta$, and let b_1, b_2 be the boxes in B_i which contain the endpoints p_1, p_2, of e. Then by Lemma 3, $(r(b_1), r(b_2)) \in X_i$, and hence $(r(b_1), r(b_2)) \in E$. We let $f(e) = \{(r(b_1), r(b_2))\}$. By Lemma 1, $d(r(b_1), r(b_2)) \leqq (1 + \varepsilon)\, d(p_1, p_2)$.

It remains to be shown that the graph $(V, \bigcup_{e \in T} f(e))$ is connected. Consider a partition of V into V_1 and V_2. Let $e = (p_1, p_2)$, where $p_1 \in V_1$ and $p_2 \in V_2$, be the unique edge in T from V_1 to V_2.

Downloaded 03/28/24 to 24.18.43.38 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

(1) Suppose $e \in (Z - Z_\delta)$. Then $f(e)$ defines a path from p_1 to p_2.

(2) Suppose $e \in (Z_i - Z_{i-1})$, $i \leqq \delta$, and boxes b_1, b_2 in B_i contain p_1, p_2, respectively. Then we must have that $(b_1 \cap V) \subseteq V_1$ and $(b_2 \cap V) \subseteq V_2$. This is seen as follows. Assume that there is a point p' in $b_1 \cap V_2$. As $d_{\min}(b_1, b_2) > d_{\max}(b_1)$, replacing (p_1, p_2) by (p_1, p') in T gives a spanning tree on V of smaller length which cannot happen. So $(b_1 \cap V) \subseteq V_1$. It follows similarly that $(b_2 \cap V) \subseteq V_2$. Then the edge $(r(b_1), r(b_2))$ connects V_1 and V_2.

We now show that E contains a linear number of edges. We note that $R_1 \subseteq R_2 \subseteq \cdots \subseteq R_i \subseteq \cdots \subseteq R_\delta$. If there is an edge in X_i between two points in R_{i-1} then the same edge is also present in X_{i-1}, and hence every edge in $X_i - X_{i-1}$ is incident on a representative point in the set of representatives $R_i - R_{i-1}$. There are at most $O((2c_a\varepsilon^{-1})^k)$ edges in X_i incident on any point in V, and so we get

$$|X| = \sum_{i=1}^{\delta} |X_i - \bigcup_{j=0}^{i-1} X_j| + |X_0| = O((2c_a\varepsilon^{-1})^k \sum_{i=1}^{\delta} (|R_i| - |R_{i-1}|))$$
$$= O((2c_a\varepsilon^{-1})^k |R_\delta|) = O((2c_a\varepsilon^{-1})^k n).$$

Then since $|Y| \leqq n$, and $E = X \cup Y$ we get that $|E| = O((2c_a\varepsilon^{-1})^k n) = O(\varepsilon^{-k} n)$ for fixed k.

To obtain a fast implementation of *Algorithm Sparse-Graph* we construct a data structure which is best described as a *tree-of-boxes*.

(1) The root of the tree is the box g_0, and the children of each box b in B_i are those boxes in B_{i+1} which are sub-boxes of b.

(2) The leaf boxes are the boxes in B_δ, and each leaf box contains a list of points in V that are in the box.

(3) The boxes at each level i, i.e., the boxes in B_i are linked together in a doubly linked list.

(4) From each box b in B_i there are pointers to (i) its father in B_{i-1}, (ii) its sons in B_{i+1}, (iii) each box b' in B_i satisfying $d_{\min}(b, b') \leqq L_i$, and (iv) the leftmost leaf box in the subtree rooted at box b.

The *tree-of-boxes* has $O(\log n)$ levels and at most n boxes per level. For each box $b \in B_i$, there are at most $O((2c_a\varepsilon^{-1})^k)$ boxes b' in B_i such that $d_{\min}(b, b') \leqq L_i$. So the *tree-of-boxes* requires $O((2c_a\varepsilon^{-1})^k n \log n)$ storage, and can be constructed in $O((2c_a\varepsilon^{-1})^k n \log n)$ time by starting from the root and proceeding towards the leaves level by level.

Once the *tree-of-boxes* is available, the representatives in boxes may be chosen in time proportional to $|B| = O(n \log n)$. For boxes b_1 and b_2, we can find points $p_1 \in b_1$, $p_2 \in b_2$, such that $d(p_1, p_2) = d_{\min}(b_1, b_2)$ in $O(k)$ time. Utilizing the *tree-of-boxes*, each of the edge sets X_i can be extracted in $O(k(2c_a\varepsilon^{-1})^k n)$ time, and so G may be extracted in $O(k(2c_a\varepsilon^{-1})^k n \log n)$ time.

Once we have the sparse graph $G = (V, E)$ one of the standard algorithms in [6], [8], [11], [15] may be used to find an MST in G. We thus have an algorithm for finding an approximate minimum spanning tree on a set V of n points in k-dimensional space. The running time is $O(\varepsilon^{-k} n \log n)$ for fixed k, and the weight of the approximate minimum spanning tree obtained is at most $(1+\varepsilon)$ times the weight of a minimum spanning tree.

4. MST algorithm with almost linear expected running time.

4.1. Description. Like the approximate minimum spanning tree algorithm in the previous section, the algorithm to find a minimum spanning tree in the complete graph on V also consists of two stages. In the first stage we extract a graph $G' = (V, E')$

Downloaded 03/28/24 to 24.18.43.38 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

which is guaranteed to contain a spanning tree of weight W_{MST}, and in the second stage we find a minimum spanning tree in G'. Unlike the graph G in the previous section, G' is not necessarily sparse; however, G' is sparse with high probability under the assumption that the given points are uniformly and independently distributed in $[0, 1]^k$.

Let g_0 be the unit k-cube $[0, 1]^k$ and let g_i be a grid that divides g_0 into 2^{ki} identical cubical boxes. Let B_i be the set of all the boxes in g_i (rather than just the set of occupied boxes) and let $B = \bigcup_{i=0}^{\delta} B_i$. Define δ to be $\lceil \log_2 n/k \rceil$. Let $L_i = c_e 2^{-i}$ where $c_e = 12.1k^{1/q}$ for L_q metric, $q = 1, 2, 3, \cdots, \infty$. Let Z be the set of all the edges in the complete graph on V and let the edge sets Z_i be defined as in § 3. We note that Lemma 2 and Lemma 3 in § 3 are still valid under these definitions.

For a box b, let $\mu(b)$ denote the singleton set containing the center of box b. For a pair of boxes b_1, b_2, we define $\pi(b_1, b_2, i)$ to be the union of those boxes b_j which are such that

(i) $b_j \in B_i$,

(ii) $d_{\max}(b_j, \mu(b_1)) < d_{\min}(b_1, b_2) - d_{\max}(b_1, \mu(b_1))$, and

(iii) $d_{\max}(b_j, \mu(b_2)) < d_{\min}(b_1, b_2) - d_{\max}(b_2, \mu(b_2))$.

We now give an algorithm to extract G'.

```
ALGORITHM PROBABLY-SPARSE-GRAPH.
1.  Let $C = \bigcup_{i=1}^{\delta} C_i$ where
        $C_i = \{(p_1, p_2): p_1 \in b_1, p_2 \in b_2, b_1 \in B_i, b_2 \in B_i, L_i/3 \leqq d_{\min}(b_1, b_2) \leqq L_i, \pi(b_1, b_2, i) \cap V = \emptyset\}$.
2.  Let $D = \{(p_1, p_2): p_1 \in b_1, p_2 \in b_2, b_1 \in B_\delta, b_2 \in B_\delta, d_{\min}(b_1, b_2) \leqq L_\delta\}$.
3.  Let $E' = C \cup D$.
end Probably-Sparse-Graph
```

We have to show that $G' = (V, E')$ contains a spanning tree which is an MST in the complete graph on V. Let T be an MST in the complete graph on V and let $e = (p_1, p_2)$ be an edge in T.

(1) Suppose $e \in (Z - Z_\delta)$. We have that $(Z - Z_\delta) \subseteq D$ and so $e \in D$.

(2) Suppose $e \in (Z_i - Z_{i-1})$, $i \leqq \delta$, and p_1, p_2 are located in boxes b_1, b_2 in B_i. We shall show that $\pi(b_1, b_2, i)$ is nonempty, so the condition $\pi(b_1, b_2, i) \cap V = \emptyset$ is not vacuously true. By Lemma 3 in § 3, $L_i/3 \leqq d_{\min}(b_1, b_2) \leqq L_i$. Let $\hat{p}$ be the center of the line segment joining the centers of boxes b_1 and b_2. It is easily shown that there is a cubical box $\hat{b}$ of size $2^{-(i-1)}$ centered at $\hat{p}$, such that $d_{\max}(\hat{b}, \mu(b_1)) < d_{\min}(b_1, b_2) - d_{\max}(b_1, \mu(b_1))$, and $d_{\max}(\hat{b}, \mu(b_2)) < d_{\min}(b_1, b_2) - d_{\max}(b_2, \mu(b_2))$. As $\hat{b}$ must contain a box in B_i we conclude that $\pi(b_1, b_2, i)$ is nonempty. Since $(p_1, p_2) \in T$, for each point p in V either $d(p, p_1) \geqq d(p_1, p_2)$ or $d(p, p_2) \geqq d(p_1, p_2)$, and it then follows that $\pi(b_1, b_2, i) \cap V$ must be empty. So $e \in C_i$.

Hence $T \subseteq C \cup D \subseteq E'$.

Algorithm Probably-Sparse-Graph can be implemented in $O(|E'| + k(2c_e)^k n)$ time by using a *tree-of-boxes*, similar to the one in § 3 for the boxes in B. We first build a skeleton for the *tree-of-boxes*. The skeleton differs from the *tree-of-boxes* described in § 3 in that there are no points in V stored at any of the leaf boxes. The skeleton has $O(n)$ boxes and can be constructed in $O((2c_e)^k n)$ time. Once the skeleton is available, we utilize a radix (bucket) sort [11], with radix $2^{k\delta}$ $(= O(n))$, to distribute the n points into the $2^{k\delta}$ leaf boxes in $O(n)$ time. We note that at this stage the *tree-of-boxes* contains all the boxes in B irrespective of whether they do or do not contain a point in V. Then in $O(n)$ time we delete from the *tree-of-boxes* all those boxes which do not contain a

Downloaded 03/28/24 to 24.18.43.38 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Downloaded 03/28/24 to 24.18.43.38 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

point in V, and add to each occupied box b a pointer to the leftmost leafbox in the subtree rooted at box b.

Utilizing the *tree-of-boxes*, we can check whether $\pi(b_1, b_2, i) \cap V$ is empty in $O(k(2c_e)^k)$ time if $b_1 \in B_i$, $b_2 \in B_i$, and $d_{\min}(b_i, b_2) \leqq L_i$. The edge set C_i may be extracted in $O(|C_i| + k(2c_e)^k n)$ time, for $i = 1, \cdots, \delta$, and D may be obtained in $O(|D| + k(2c_e)^k n)$ time. So *Algorithm Probably-Sparse-Graph* may be implemented in $O(|E'| + k(2c_e)^k n)$ time.

We shall now describe a procedure for finding an MST in the graph $G' = (V, E')$. We first try to sort the edges in G' by employing a fixed number of passes of radix (bucket) sort [11], with radix (base) $2^{\lceil \log_2 n \rceil}$. If this approach fails to sort the edges in G' we sort them in $O(|E'| \log |E'|)$ time using a standard algorithm [11]. After sorting the edges, we utilize Kruskal's algorithm [9], [11] to find an MST in G'. The edges are examined in increasing order of weight and an edge is chosen if and only if it does not form a cycle with the previously chosen edges. The chosen edges form an MST in G'. Once a sorted list of edges is available, Kruskal's algorithm may be implemented in $O(|E'|\alpha(|E'|, n))$ time, where α is the inverse Ackermann function defined in [13].

The algorithm to find an MST in the complete graph on V consists of first extracting G' using *Algorithm Probably-Sparse-Graph*, and then finding an MST in G' using the above described procedure. Let us assume that V consists of n random points, independently and uniformly distributed in the unit k-cube $[0, 1]^k$. Let P_L be the probability that for some i, $0 \leqq i \leqq kn^7 - 1$, the lengths of at least two edges lie in the interval $[in^{-7}, (i+1)n^{-7}]$. In § 4.2 we show that P_L is $o(1/n^2)$. So the probability that eight passes of radix sort, with radix $2^{\lceil \log_2 n \rceil}$, fail to sort the edges in G' is $o(1/n^2)$; thus the probability that the edges in G' cannot be sorted in $O(|E'|)$ time, using a fixed number of passes of radix sort, is $o(1/n^2)$. Let $P_{G'}$ be the probability that G' contains more than cn edges, where c is a constant dependent on the dimension k. In § 4.2 we also show that $P_{G'}$ is $o(1/n^2)$. So with probability $1 - o(1/n^2)$, the running time of the algorithm to find an MST in the complete graph on V is $O(cn\alpha(cn, n) + k(2c_e)^k n)$. The worst case running time of the same algorithm is $O(n^2 \log n)$. Therefore, when k is fixed, the expected running time of the algorithm for finding an MST in the complete graph on V is $O(n\alpha(cn, n))$.

4.2. Probabilistic analysis. We assume that the set V consists of n random points which are independently and uniformly distributed in the unit k-cube $[0, 1]^k$. Let P_L be the probability that for some i, $0 \leqq i \leqq kn^7 - 1$, the lengths of at least two edges lie in the interval $[in^{-7}, (i+1)n^{-7}]$. Let $P_{G'}$ be the probability that G' contains more than cn edges, where c is a constant dependent on the dimension k. We have to show that both $P_{G'}$ and P_L are $o(1/n^2)$.

We shall first bound P_L. Fix an ordering on the points in the unit k-cube. Assume that there are at least two edges e_1 and e_2 whose lengths lie in the interval $[in^{-7}, (i+1)n^{-7}]$, for some i. Suppose e_1, e_2 have an endpoint in common and $e_1 = (x, y)$, $e_2 = (y, z)$. Then the points x and z are restricted to lie in a shell, of thickness n^{-7} and radius at most k, around point y, and there are at most n^3 possibilities for triples (x, y, z). Suppose e_1, e_2 do not have an endpoint in common and $e_1 = (x, y)$, $e_2 = (z, w)$. Without loss of generality let $x \leqq y$ and $z \leqq w$. Then x is restricted to lie in a shell, of thickness n^{-7} and radius at most k, around y; z must lie in a similar shell around w; and there are at most n^4 choices for tuples (x, y, z, w). We thus have

$$P_L \leqq kn^7\left(\left(\frac{c_1}{n^7}\right)^2 n^3 + \left(\frac{c_2}{n^7}\right)^2 n^4\right) = o\left(\frac{1}{n^2}\right)$$

where c_1 and c_2 are constants dependent on dimension k.

We shall now estimate $P_{G'}$. Let us assume that n is a power of 2^k. Let $m_i = 2^{ki}$,

$$\beta_i = \{b \colon b \in B_i, (p_1, p_2) \in C_i, \text{ either } p_1 \in b \text{ or } p_2 \in b\},$$

and

$$\Pi_i = \{\pi(b_1, b_2, i) \colon b_1 \in B_i, b_2 \in B_i, p_1 \in b_1, p_2 \in b_2, (p_1, p_2) \in C_i\}.$$

We note that the volume of each region in Π_i is at least m_i^{-1}. There exists a constant $c_5 \geqq 1$, dependent on k, such that for each i, $i = 1, \cdots, \delta$, each region in Π_i intersects at most c_5 other regions in Π_i. Let $c_3 = (36k+3)^k$. There are at most $c_3 m_i$ possibilities for regions in Π_i. We let c_6 be the smallest positive integer greater than 16 such that $2^{c_6}/c_5 > 2 + \log_e (c_3 c_5) + 6c_6$ and $c_6 \geqq 1 + 2^{c_6} \log_e (1 + 2^{-c_6})$.

We define P_i as follows.

(1) For $n/m_i > 6c_3 \log_2 n$, let P_i be the probability that C_i is not empty.

(2) For $2^{c_6} < n/m_i \leqq 6c_3 \log_2 n$, let P_i be the probability that C_i contains more than $c_3 m_i$ edges.

(3) For $1 \leqq n/m_i \leqq 2^{c_6}$, let P_i be the probability that the number of edges in C_i exceeds $c_3 c_7 m_i$ where $c_7 = 2e^4 2^{2c_6}$.

Let P_D be the probability that the number of edges in D exceeds $c_3 c_7 n$.

Let the constant c in the definition of $P_{G'}$ be $4c_3c_7$. If the number of edges in G' exceeds $4c_3c_7n$ then either D contains more than c_3c_7n edges, or for some i, $1 \leqq i \leqq \delta$, C_i contains more than $c_3c_7m_i$ edges. Hence,

$$P_{G'} \leqq \sum_{i=1}^{\delta} P_i + P_D.$$

We shall show that each of P_i and P_D is $o(1/n^4)$ and it then follows that $P_{G'} = o(1/n^2)$ as $\delta \leqq \log_2 n$. In order to bound P_i we consider three cases depending on the value of n/m_i.

Case 1. $n/m_i > 6c_3 \log_2 n$. In this case we show that there is a large empty region in the unit k-cube $[0, 1]^k$. If C_i is nonempty there is at least one region in Π_i and this region does not contain a point in V. There are at most $c_3 m_i$ possibilities for regions in Π_i and each of these regions has volume at least $(m_i)^{-1}$. Hence,

$$P_i \leqq c_3 m_i \left(1 - \frac{1}{m_i}\right)^n = o\left(\frac{1}{n^4}\right).$$

We shall now give a lemma that will be useful for Case 2 and Case 3. Let P_{ij} be the probability that the number of boxes in B_i, which contain at least $e^2 2^j$ and at most $e^2 2^{j+1}$ points in V, exceeds $m_i/2^{3j}$.

LEMMA 4. *Let* $2^{j_0} \geqq (n/m_i) \geqq 1$, *and* $(n/m_i) \leqq 6c_3 \log_2 n$. *Then* $\sum_{j=j_0}^{\log_2 n} P_{ij} = o(1/n^4)$.

Proof. Suppose $2^j \geqq 6c_3 \log_2 n$, and $2^j \geqq (n/m_i)$. As P_{ij} is bounded by the probability that there is at least one box in B_i with at least $e^2 2^j$ points, we have

$$\begin{aligned} P_{ij} &\leqq m_i \binom{n}{e^2 2^j} (m_i)^{-e^2 2^j} \\ &\leqq m_i \frac{n^{e^2 2^j}}{(e^2 2^j)!} (m_i)^{-e^2 2^j} \\ &\leqq O\left(m_i \left(\frac{n}{m_i 2^j}\right)^{e^2 2^j} e^{-e^2 2^j} \right) \quad \text{(using Stirling's approx.)} \\ &= o\left(\frac{1}{n^5}\right). \end{aligned}$$

Downloaded 03/28/24 to 24.18.43.38 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

Suppose $1 \leqq (n/m_i) \leqq 2^j < 6c_3 \log_2 n$. We can get a bound on P_{ij} by first choosing $m_i/2^{3j}$ boxes in B_i, then choosing $e^2 m_i/2^{2j}$ points to be located in the chosen boxes and letting the remaining points lie anywhere in the unit k-cube. So

$$P_{ij} \leqq \binom{m_i}{m_i/2^{3j}} \binom{n}{e^2 m_i/2^{2j}} 2^{-3je^2 m_i 2^{-2j}}$$

$$\leqq \frac{(m_i)^{m_i 2^{-3j}}}{(m_i 2^{-3j})!} \frac{n^{e^2 m_i 2^{-2j}}}{(e^2 m_i 2^{-2j})!} 2^{-3je^2 m_i 2^{-2j}}$$

$$\leqq O\left((e2^{3j})^{m_i 2^{-3j}} e^{-e^2 m_i 2^{-2j}} \left(\frac{n}{m_i 2^j}\right)^{e^2 m_i 2^{-2j}} \right) \quad \text{(Stirling's approx.)}$$

$$= o\left(\frac{1}{n^5}\right).$$

The proof of the lemma then follows. □

Case 2. $2^{c_6} < n/m_i \leqq 6c_3 \log_2 n$. This case is broken down into two subcases depending on the number of boxes in β_i. We have

$$P_i \leqq Pr(|\beta_i| > 8m_i^7 n^{-6}) + Pr(|\beta_i| \leqq 8m_i^7 n^{-6} \text{ and } |C_i| > c_3 m_i)$$
$$= o(1/n^4) + o(1/n^4) = o(1/n^4).$$

If $|\beta_i| > 8m_i^7 n^{-6}$ then there is a large empty region, whereas if $|\beta_i| \leqq 8m_i^7 n^{-6}$ and $|C_i| > c_3 m_i$ then there is a concentration of points in some region.

Case 2.1. Suppose $|\beta_i| > 8m_i^7 n^{-6}$. There are at least $|\beta_i|/2$ regions in Π_i and as a region in Π_i intersects at most c_5 other regions in Π_i, we can find at least $c_5^{-1} m_i^7 n^{-6}$ disjoint regions in Π_i each of which does not contain any point in V. As there are at most $c_3 m_i$ choices for regions in Π_i, and each region in Π_i has volume at least m_i^{-1}, we have

$$Pr(|\beta_i| > 8m_i^7 n^{-6}) \leqq \binom{c_3 m_i}{c_5^{-1} m_i^7 n^{-6}} (1 - c_5^{-1} m_i^6 n^{-6})^n$$

$$\leqq \frac{(c_3 m_i)^{c_5^{-1} m_i^7 n^{-6}}}{(c_5^{-1} m_i^7 n^{-6})!} (1 - c_5^{-1} m_i^6 n^{-6})^n.$$

Using Stirling's approximation for factorials, taking logarithms and noting that $\log_e (1 - x) \leqq -x$, for $0 \leqq x < 1$, we get

$$\log_e (Pr(|\beta_i| > 8m_i^7 n^{-6})) \leqq nc_5^{-1} \left(\frac{m_i}{n}\right)^6 \left(-1 + \frac{m_i}{n}\left(1 + \log_e (c_3 c_5) + 6 \log_e \left(\frac{n}{m_i}\right)\right)\right)$$

$$\leqq -\frac{n}{c_5 2^{c_6} (6c_3 \log_2 n)^6}.$$

Thus

$$Pr(|\beta_i| > 8m_i^7 n^{-6}) = o(1/n^4).$$

Case 2.2. Suppose $|\beta_i| \leqq 8m_i^7 n^{-6}$ and $|C_i| > c_3 m_i$. Let $j_0 = 2 \log_2 (n/m_i) - 1$. A point in any box b in B_i is joined to points in at most c_3 other boxes in B_i by edges in C_i, and so we have that

$$c_3 \sum_{b \in \beta_i} (|b \cap V|)^2 \geqq |C_i| > c_3 m_i$$

Downloaded 03/28/24 to 24.18.43.38 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

and hence

$$\sum_{b\in\beta_i,\,|b\cap V|\geqq 2^{j_0}} (|b\cap V|)^2 > (1-2^{-c_6+1})m_i.$$

Then for some $j, j=j_0(=2\log_2(n/m_i)-1), \cdots, \log_2 n$, the number of boxes in β_i (and hence B_i), which contain at least $e^2 2^j$ and at most $e^2 2^{j+1}$ points in V, exceeds $m_i/2^{3j}$. Then from Lemma 4,

$$Pr(|\beta_i|\leqq 8m_i^7 n^{-6} \text{ and } |C_i|>c_3 m_i)\leqq \sum_{j=j_0}^{\log_2 n} P_{ij} = o\left(\frac{1}{n^4}\right).$$

Case 3. $1\leqq n/m_i\leqq 2^{c_6}$. Suppose $|C_i|>c_3c_7m_i$. A point in box b in B_i is joined to points in at most c_3 other boxes in B_i by edges in C_i, and so we get

$$c_3\sum_{b\in B_i}(|b\cap V|)^2\geqq |C_i|>c_3c_7m_i=2e^4 2^{2c_6}c_3m_i;$$

hence,

$$\sum_{b\in B_i,\,|b\cap V|\geqq 2^{c_6}} (|b\cap V|)^2 > (2e^4-1)2^{2c_6}m_i.$$

Then for some $j, j=c_6, \cdots, \log_2 n$, the number of boxes in B_i, which contain at least $e^2 2^j$ points and at most $e^2 2^{j+1}$ points in V exceeds $m_i/2^{3j}$. Then from Lemma 4 we can conclude that

$$P_i\leqq \sum_{j=c_6}^{\log_2 n} P_{ij} = o\left(\frac{1}{n^4}\right).$$

By reasoning in the same manner as in Case 3 above we can show that $P_D = o(1/n^4)$.

5. Conclusions. We have given an $O(\varepsilon^{-k} n\log n)$ algorithm for finding an approximate minimum spanning tree on a set V of n points in k-dimensional space, the weight of the approximate minimum spanning tree is guaranteed to be at most $(1+\varepsilon)$ times the weight of a minimum spanning tree. We have also presented an algorithm for finding a minimum spanning tree on a set V of n points in k-dimensional space. Under the assumption that the set V consists of n random points, independently and uniformly distributed in the unit k-cube $[0, 1]^k$, the expected running time of this minimum spanning tree algorithm is shown to be $O(n\alpha(cn, n))$ where c is a constant dependent on k and α is the inverse Ackermann function.

Acknowledgments. The author would like to thank Professor C. L. Liu, S. Kapoor, and P. Ramanan for helpful discussions.

REFERENCES

[1] C. Berge and A. Ghouila-Houri, *Programming, Games and Transportation Networks*, John Wiley, New York, 1965.

[2] K. Clarkson, *Fast expected-time and approximation algorithms for geometric minimum spanning trees*, Proc. 16th Annual ACM Symposium on the Theory of Computing, 1984, pp. 342-348.

[3] D. Cheriton and R. E. Tarjan, *Finding minimum spanning trees*, this Journal, 5 (1976), pp. 724-742.

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

[5] R. O. Dude and P. E. Hart, *Pattern Classification and Science*, John Wiley, New York, 1973.

[6] M. L. Fredman and R. E. Tarjan, *Fibonacci heaps and their uses in network optimization algorithms*, Proc. 25th Annual IEEE Symposium on the Foundations of Computer Science, 1984, pp. 338-346.

[7] H. Gabow, J. Bentley, and R. Tarjan, *Scaling and related techniques for geometric problems*, Proc. 16th Annual ACM Symposium on the Theory of Computing, 1984, pp. 135-143.

Downloaded 03/28/24 to 24.18.43.38 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy

[8] H. N. GABOW, Z. GALIL, AND T. H. SPENCER, *Efficient implementation of graph algorithms using contraction*, Proc. 25th Annual IEEE Symposium on the Foundations of Computer Science, 1984, pp. 347-357.

[9] J. B. KRUSKAL, *On the shortest spanning subtree of a graph and the travelling salesman problem*, Amer. Math. Soc., 7 (1956), pp. 48-50.

[10] R. C. PRIM, *Shortest connection networks and some generalizations*, Bell System Tech. J., 36, pp. 1388-1401.

[11] E. M. REINGOLD, J. NIEVERGELT, AND N. DEO, *Combinatorial Algorithms: Theory and Practice*, Prentice-Hall, Englewood Cliffs, NJ.

[12] M. I. SHAMOS AND D. J. HOEY, *Closest-point problems*, Proc. 16th Annual IEEE Symposium on the Foundations of Computer Science, 1975, pp. 151-162.

[13] R. E. TARJAN, *Efficiency of good but not linear set union algorithm*, J. Assoc. Comput. Mach., 22 (1975), pp. 215-225.

[14] P. M. VAIDYA, *A fast approximation algorithm for minimum spanning trees in k-dimensional space*, Proc. 25th Annual IEEE Symposium on the Foundations of Computer Science, 1984, pp. 403-407.

[15] A. C. YAO, *A $O(|E||\log\log V|)$ algorithm for finding minimum spanning trees*, Inform. Process. Lett., 4 (1975), pp. 21-23.

[16] A. C. YAO, *On constructing minimum spanning trees in k-dimensional spaces and related problems*, this Journal, 11 (1982), pp. 721-736.

[17] C. T. ZAHN, *Graph-theoretical method for detecting gestalt clusters*, IEEE Trans. Comput., C-20 (1970), pp. 68-70.

Downloaded 03/28/24 to 24.18.43.38 . Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/terms-privacy