A fast approximation algorithm for minimum spanning trees in k-dimensional space

Pravin M. Vaidya

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801

Abstract

We study the problem of finding a minimum spanning tree on the complete graph on n points in E^k, with the weight of an edge between any two points being the distance between the two points under some distance metric. A fast algorithm, which finds an approximate minimum spanning tree with weight at most $(1+\epsilon)$ times optimal in $O(n \log n\,((\log n)^k + \log(\epsilon^{-1})(\log n)^{k-1}\epsilon^{-(k-1)}))$ time, is developed for the L_q, $q=2,3,...$, distance metrics. Moreover, if the n points are assumed to be independently and uniformly distributed in the box $[0,1]^k$, then the probability that the approximate minimum spanning tree found is an exact minimum spanning tree is shown to be $(1-o(1/n))$.

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 an MST were given by Dijkstra [5], Kruskal [8], Prim [10] and Sollin [2]. 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 [14] that an MST may be found in $O(m \text{ loglog } n)$ time. Further results on MST's may be found in Cheriton and Tarjan [4].

A set of n points in k-dimensional space can be thought of as the set of vertices of a complete undirected graph, with the weight of an edge between any two points being the distance between the points under some distance metric. Each point x is given as a vector $(x_1,x_2,...,x_k)$. We use E^k to denote the space of all k-tuples of real numbers, and E_q^k, $q=1,2,...,\infty$, to denote the space of all k-tuples of real numbers with L_q metric, i.e. the distance between any two points x and y is given by $d_q(x,y) = (\sum_{i=1}^{k} |x_i - y_i|^q)^{\frac{1}{q}}$. (Note that $d_\infty(x,y) = \max_i |x_i - y_i|$). A simple way to find an MST on such a graph is to explicitly compute all the edge weights and use an $O(n^2)$ algorithm for general graphs. Shamos and Hoey [12] gave an $O(n \log n)$ algorithm for n points in the plane ($k=2$) with Euclidean metric. Yao [15] gave algorithms which construct an MST in time $O(n^{2-2^{-(k+1)}}(\log n)^{1-2^{-(k+1)}})$ for any fixed $k \geq 3$, and the distance metrics L_q, $q=1,2,\infty$. Fast algorithms for fixed $k \geq 2$ and the L_1 and L_∞ distance metrics are given in [7].

*This work was partially supported by the National Science Foundation under grant MCS 81-07647.

In many applications of MST's like clustering and pattern recognition [6, 17], a spanning tree whose weight is close to the weight of an MST would serve the purpose just as well. Also, in geometric and statistical applications an approximate minimum spanning tree (AMST) might be adequate for the job. So it is useful to investigate if there exist fast algorithms for finding an AMST, with the property that the weight of the AMST obtained is at most $(1+\epsilon)$ times the weight of an MST, for any given fixed $\epsilon>0$. An algorithm for finding an AMST for $k=3$ and the L_2 distance metric is given in [3] but the running time depends on the ratio of the maximum to the minimum distance between any two points. We develop an algorithm for constructing an AMST on the associated complete graph on a given set of n points in E_q^k, where $q=2,3,4,.....$, which runs in $O(n \log n\,((\log n)^k + \log(\epsilon^{-1})(\log n)^{k-1}\epsilon^{-(k-1)}))$ time, and is guaranteed to produce an AMST whose weight is at most $(1+\epsilon)$ times the weight of an MST, for any fixed $\epsilon>0$. We also show that for n random points, independently and uniformly distributed in the box $[0,1]^k$, the probability that the AMST found by the algorithm is an exact MST is $(1-o(1/n))$.

Similar techniques can be used to devlelop algorithms for finding an MST under the L_1 and L_∞ distance metrics with running times $O(n(\log n)^{k+2})$ and $O(n(\log n)^{k+1})$, respectively, but as fast algorithms for the L_1 and L_∞ distance metrics have already appeared in [7] we shall not describe them here.

As far as the model of computation is concerned, we assume a random access machine with arithmetic on real numbers and charge uniform cost for all access, arithmetic and comparison operations.

We introduce a few definitions before proceeding further. All definitions pertain to some E_q^k, $q=1,2,3,...$ and some fixed $\epsilon>0$. An ϵ-closest neighbor in S_2 of a point p_1 in S_1 is some point p_2 in S_2 such that $length((p_1,p_2)) \leq (1+\epsilon)\,length((p_1,p'))$, where p' is a closest neighbor of p_1 in S_2. An ϵ-smallest edge from S_1 to S_2 is an edge e from a point in S_1 to a point in S_2 such that $length(e) \leq (1+\epsilon)\,length(e')$, where e' is the smallest edge from a point in S_1 to a point in S_2. An r-neighborhood of a point p is the set of those points whose distance from p is less than or equal to r under the distance measure under consideration.

0272-5428/84/0000/0403$01.00 © 1984 IEEE

2. The approximate minimum spanning tree algorithm

The approximate spanning tree algorithm follows the skeleton given below.

Algorithm AMST

Begin

Step 1. Build a range tree [1, 9, 13] on the given set S of n points, and form a collection of k ordered lists $List_i$, $i=1,2,...,k$, with $List_i$ containing all the points in S sorted on the i^{th} co-ordinate. Initialize the forest to consist of trees with a single node each and initialize the list *Tree-edges* to *nil*.

Step 2. Repeat steps 2.1 through 2.5 till the the forest contains a single tree.

Step 2.1. Pick T, a tree containing the least number of points.

Step 2.2. Delete all points in T from the data structures being used and initialize the list of *candidate-edges*.

Step 2.3. Using the data structures which now contain only points belonging to $(S-T)$, for each point p in T, under the required distance measure, find an ϵ-closest neighbor p' in $(S-T)$ and add edge (p,p') to the list of *candidate-edges*.

Step 2.4. The smallest edge e on the list of *candidate-edges* is an ϵ-smallest edge between T and $S-T$. Let T and T' be the trees joined together by edge e. Coalesce T and T', thereby reducing the number of trees in the forest by one and add e to the list of *tree-edges*.

Step 2.5. Restore all data structures by inserting back all points in tree T.

end *AMST*

The routine *Algorithm ϵ-closest neighbor* for finding an ϵ-closest neighbor in $(S-T)$ of a point p in T is described in Section 3. A brief sketch of this routine is as follows. The routine maintains two distances r_1 and r_2 such that the r_1-neighborhood of p does not contain a point in $(S-T)$ whereas the r_2-neighborhood of p does contain a point in $(S-T)$. Let $r=(r_1+r_2)/2$. The r-neighborhood of p is approximated by a collection of boxes (with sides parallel to the co-ordinate axes) such that the collection of approximating boxes contains the r-neighborhood of p and is itself contained in the $r(1+\epsilon')$-neigborhood of p, for some ϵ' dependent on ϵ. By means of a sequence of range searches it is determined whether the approximating boxes do or do not contain a point in $(S-T)$, and r_1 and r_2 are suitably updated depending on the outcome of the range search. We stop when r_1 and r_2 are close enough.

The routine *Algorithm Genboxes* for generating a collection of boxes which approximates the r-neighborhood of a point p is described in Section 4. A brief overview is as follows. The r-neighborhood of p is projected onto a sequence of planes perpendicular to one of the co-ordinate axes. The projection of the r-neighborhood onto a plane itself forms a smaller neighborhood of the projection of p onto the plane, in a space of one less dimension. A collection of boxes which approximates the projection of the r-neighborhood onto a plane is recursively obtained and then used to generate a collection of boxes which approximates the portion of the r-neighborhood between this plane and the next plane in the sequence.

The probability that the AMST found is an exact MST, under the assumption that the n points are independently and uniformly distributed in $[0,1]^k$, is computed in Section 5.

Utilising a range tree [1, 9, 13], we can search a parallelepiped in E^k in $O((\log n)^{k-1})$ time. Building the range tree intially takes $O(n(\log n)^{k-1})$ time and all points in some subset T of S may be inserted into or deleted from the range tree in $O(|T|(\log n)^{k-1})$ time. Each list in the collection $List_i$, $i=1,2,...,k$, is implemented as a 2-3 tree, with each node containing three extra units of information giving the number of elements in the left, middle, and right subtrees rooted at the node. This allows the operations of insertion, deletion, searching for some element, and accessing the element with some given rank j, to be done in $O(\log n)$ time.

The forest maintenance may be efficiently carried out using *UNION* and *FIND* procedures described in [11] and this requires time $O(n \log n)$. A tree containing the least number of points may be picked by maintaining a priority queue for the trees in the forest and the total work for maintaining the queue is bounded by $O(n \log n)$. An ϵ-smallest edge between tree T and $(S-T)$ may be found in $O(|T|((\log n)^k + \log(\epsilon^{-1})(\log n)^{k-1}\epsilon^{-(k-1)}))$ time. As at each stage the tree with the least number of points is chosen, the overall running time of the algorithm is $O(n \log n((\log n)^k + \log(\epsilon^{-1})(\log n)^{k-1}\epsilon^{-(k-1)}))$.

We now show that the AMST generated by the algorithm has weight at most $(1+\epsilon)$ times the weight of an MST. Let T_m be an MST and let T_a be the AMST produced by *Algorithm AMST*; we shall give a bijection $f: T_a \to T_m$ such that for any e in T_a, $length(e) \leq (1+\epsilon)length(f(e))$. As the algorithm proceeds we shall carry around a subset of edges in T_m. Let Z_i be the set of edges in T_m still remaining after i edges have been added to the AMST T_a. A correspondence will be drawn between the edge deleted from Z_i to obtain Z_{i+1} and the $(i+1)^{st}$ edge added to T_a.

Claim. The edges in Z_i form a spanning tree on the forest at stage i, if we consider each tree in the forest to be a super node.

The claim holds in the beginning as every tree in the forest consists of a single node and as $Z_0 = T_m$. Let e be the $(i+1)^{st}$ edge added to T_a and let T and T' be the two trees joined together by e. There are two cases.

Case 1. There is an edge l in Z_i between T and T'. Then as e is an ϵ-smallest edge from T to $(S-T)$, we have $length(e) \leq (1+\epsilon)length(l)$. We let $f(e)=l$ and obtain $Z_{i+1} = Z_i - \{l\}$. Z_{i+1} still forms a spanning tree on the forest after T and T' are coalesced.

Case 2. There is no edge in Z_i between T and T'. From the above claim, there is a path from T to T'. Let l be the first edge on this path; l goes between tree T and some other tree T''; moreover there is an edge in Z_i, other than l, incident on T''. We let $f(e) = l$ and $Z_{i+1} = Z_i - \{l\}$. Z_{i+1} will form a spanning tree on the forest after merging trees T and T'. Also as e is an ϵ-smallest edge between T and $(S-T)$, we have $length(e) \leq (1+\epsilon)\, length(f(e))$.

This completes the proof of the existence of the required bijection f between the AMST T_a and some fixed MST T_m.

3. Finding an ϵ-closest neighbor

In this section we describe a procedure *Algorithm ϵ-closest neighbor* for finding an ϵ-closest neighbor in $(S-T)$ of a point p in T, under L_q, $q=2,3,\ldots$. distance metrics. The algorithm takes as input a range tree containing all the points in $(S-T)$; a subset T of S; a point p in T; ϵ, the measure of closeness; q, the distance metric index; and $List_i$, containing all the points in $(S-T)$ sorted on the i^{th} co-ordinate, for $i=1,2,\ldots,k$.

The routine maintains two distances r_1 and r_2 such that the r_1-neighborhood of p, under the L_q distance measure, does not contain a point in $(S-T)$ whereas the r_2-neighborhood of p, under the L_q distance measure, does contain a point in $(S-T)$. A sample point p' in the r_2-neighborhood of p is also maintained. At each stage, the r-neighborhood of p, where $r=(r_1+r_2)/2$, is approximated by a collection of $O(\epsilon^{-(k-1)})$ boxes such that the collection of boxes contains the r-neighborhood of p and is itself contained in the $r(1+\epsilon')$-neighborhood of p, where $\epsilon' = \epsilon/(4+2\epsilon)$. The approximating boxes are generated using *Algorithm Genboxes* described in Section 4. By a sequence of range searches (utilising the range tree), it is determined whether the approximating collection of boxes does or does not contain a point in $(S-T)$. r_1 is set to r if the boxes do not contain a point in $(S-T)$. On the other hand if the collection of boxes does contain a point in $(S-T)$, r_2 is set to $d_q(p,p')$ where p' is computed as follows. If the approximating boxes contain less than $(\epsilon^{-(k-1)}(\log n)^{k-1})$ points in $(S-T)$ then we let p' be a point in $(S-T)$ that is located in the approximating boxes and is closest to p among all points in $(S-T)$ located in the approximating boxes; otherwise we let p' be some arbitrary point in $(S-T)$ that is located in the approximating boxes. As the boxes contain the r-neighborhood of p and are themselves contained in the $r(1+\epsilon')$-neighborhood of p, the properties associated with r_1 and r_2 are preserved by the update.

An initial estimate of r_1 and r_2 is obtained by first finding α, the distance between p and its closest L_∞-neighbor p_∞ in $(S-T)$, and letting $r_1 = (\alpha/k) - \epsilon$ and $r_2 = \alpha$. α may be determined by a sequence of binary searches in $O((\log n)^k)$ time, utilising the range tree and the collection of ordered lists $List_i$, $i=1,2,\ldots,k$. We start with $p' = p_\infty$.

The algorithm stops when $r_2 \leq (r_1(1+2\epsilon'))/(1-2\epsilon')$. p' is then an ϵ-closest neighbor in $(S-T)$ of p. Moreover, we are guaranteed that if p' is not a closest neighbor of p in $(S-T)$ then there are at least $(\epsilon^{-(k-1)}(\log n)^{k-1})$ points in $(S-T)$ such that the distance of each of these points from p is at least $d_q(p,p')/(1+2\epsilon)$ and at most $(1+2\epsilon)\, d_q(p,p')$.

An update of r_1 or r_2 decreases the difference $(r_2 - r_1)$ by at least a factor of $4/3$ and so the number of stages in the algorithm is bounded by $O(\log k \log(\epsilon^{-1}))$. The time for searching the approximating boxes using the range tree is bounded by $O((\log n)^{k-1})$ for every stage. The time for generating the boxes is $O(\epsilon^{-(k-1)})$. Moreover, a closest L_∞-neighbor in $(S-T)$ of a point p in T may be found in time $O((\log n)^k)$. This gives a bound of $O((\log n)^k + \log(\epsilon^{-1})(\log n)^k \epsilon^{-(k-1)})$ for the overall running time of the algorithm.

4. Generation of approximating boxes

In this section we describe how to generate the collection of boxes that approximates the r-neighborhood of some point p. The projection of any point w in the r-neighborhood of p onto the $x_k = p_k + z$ and $x_k = p_k - z$, $z>0$, planes satisfies $(\sum_{i=1}^{k-1} |w_i - p_i|^q) \leq r^q - z^q$. So the projections of the r-neighborhood of p, onto the $x_k = p_k + z$ and $x_k = p_k - z$ planes, themselves form $(r^q - z^q)$-neighborhoods in a space of dimension $(k-1)$, of the projections of p onto the respective planes. Suppose we have a collection of boxes that contains and approximates the $(r^q - z^q)$-neighborhood of the projection of p onto the $x_k = p_k + z$ plane. Each of these boxes is defined by a set of $(k-1)$ closed intervals which give an upper and a lower bound on each of the first $(k-1)$ co-ordinates of a point in E^k. Then, adding the constraint $x_k \in [p_k + z, p_k + z + \delta]$ to the set of constraints defining each box in the collection, produces a collection of closed boxes in E^k which contain and approximate the portion of the r-neighborhood of p which lies between the $x_k = p_k + z$ and $x_k = p_k + z + \delta$ planes. On the other hand, adding the constraint $x_k \in [p_k - z, p_k - z - \delta]$ to the set of constraints coresponding to each box in the collection, gives a set of boxes which contain and approximate the part of the r-neighborhood of p which lies between the $x_k = p_k - z$ and $x_k = p_k - z - \delta$ planes. This gives a way of recursively contructing the desired collection of boxes. We take the projection of the r-neighborhood of p onto each plane in a set of planes given by $x_k = p_k + ri\epsilon$, $i=0,1,\ldots,\lceil \epsilon^{-1} \rceil - 1$. For each i, the projection of the r-neighborhood onto the $x_k = p_k + ri\epsilon$ plane is recursively approximated, and the boxes approximating this projection are utilised to obtain a collection of boxes which approximate the portions of the r-neighborhood, between the $x_k = p_k + ri\epsilon$ and $x_k = p_k + r(i+1)\epsilon$ planes, and between the $x_k = p_k - ri\epsilon$ and $x_k = p_k - r(i+1)\epsilon$ planes.

Algorithm Genboxes

Input: p, the point around which the set of boxes is to be generated; r, size of the neighborhood of p to be approximated; k, the dimension; L_q, the distance measure; ϵ, the measure of accuracy, $0 < \epsilon \leq 1$.

Output: a collection of boxes which contains the r-neighborhood of p and is itself contained in the $(r(1+2(k-1)\epsilon))$-neighborhood of p.

Begin

If $k=1$ then return a list containing two one-dimensional boxes defined by $[p_i - r, p_i]$ and $[p_i, p_i + r]$;

else

Begin

Compute an r' satisfying $r(1-i^q\epsilon^q)^{1/q} \leq r' \leq r(1-i^q\epsilon^q)^{1/q}(1+\epsilon/4k)$;

List-of-boxes := *nil*;

For $i=0$ to $\left(\left\lceil \epsilon^{-1} \right\rceil - 1\right)$ do

Begin

Call *Genboxes* recursively to obtain a list L of boxes which contain and approximate the r'-neighborhood of the projection of p onto the $x_k = p_k + ri\epsilon$ plane, in a space of one less dimension;

For every box b in L add constraints $x_k \in [p_k + ri\epsilon, p_k + r(i+1)\epsilon]$ and $x_k \in [p_k - ri\epsilon, p_k - r(i+1)\epsilon]$ to the set of constraints defining box b, in order to get two sets of constraints, each set defining a box in E^k; and put the two boxes so defined on *List-of-boxes*;

end;

end;

return *List-of-boxes*;

end *Genboxes*.

The floor and ceiling of $1/\epsilon$ may be computed in $O(\log(\epsilon^{-1}))$ time. Moreover, if ϵ is fixed the qth roots of the quantities $(1-i^q\epsilon^q)$, $i=0,\ldots,(\epsilon^{-1}-1)$, may be computed once and for all, and stored in an array for repeated use by *Algorithm Genboxes*. So $\lceil 1/\epsilon \rceil$ such computations are sufficient. Since the quantity Q whose qth root is to computed is always greater than or equal to ϵ, we may first compute j such that $(1+1/q)^{(j-1)q} \leq 1/Q \leq (1+1/q)^{jq}$ in $O(\log(1/\epsilon^{-1}))$ steps and then use the Newton-Raphson method with $(1+1/q)^{-j}$ as an intial approximation to the q^{th} root of Q to obtain r' satisfying $Q^{1/q} \leq r' \leq Q^{1/q}(1+\epsilon/4k)$ in $O(\log(\epsilon^{-1}))$ iterations. Once the precomputed roots are available the algorithm takes time proportional to the number of boxes generated which is $O(\epsilon^{k-1})$.

That the collection of approximating boxes contains the r-neighborhood of p is evident from the construction. What remains to be shown is that the maximum L_q distance between p and any point in any of the boxes is at most $r(1+2(k-1)\epsilon)$. Assume that for any dimension $i \leq (k-1)$, the procedure generates a set of boxes such that the maximum distance d_i between p and a point in any box satisfies $d_i^q \leq r^q(1+2(i-1)\epsilon)$. This may be easily verified for $(k-1)=1$. By hypothesis, the maximum distance d_{k-1} between the projection of p onto the $x_k = p_k + ri\epsilon$ plane and any point in the boxes approximating the $(r(1-\epsilon^q i^q)^{1/q})$- neighborhood of the projection of p onto the $x_k = p_k + ri\epsilon$ plane, in a space of dimension $(k-1)$, satisfies $d_{k-1}^q \leq r^q(1-i^q\epsilon^q)(1+\epsilon/4k)^q(1+2(k-2)\epsilon)^q$. The maximum distance d_k between p and any point in the approximating boxes lying between the $x_k = p_k + ri\epsilon$ and $x_k = p_k + r(i+1)\epsilon$ planes or the $x_k = p_k - ri\epsilon$ and $x_k = p_k - r(i+1)\epsilon$ planes satisfies $d_k^q \leq d_{k-1}^q + r^q(i+1)^q\epsilon^q$. These two relations together with a little algebraic manipulation and the facts $i\epsilon \leq 1$ and $\epsilon \leq 1$ lead to the desired relation $d_k^q \leq r^q(1+2(k-1)\epsilon)^q$.

5. Probabilistic Analysis

Consider a set S of n random points independently and uniformly distributed in the box $[0,1]^k$. We shall compute an upper bound on the probability that the AMST generated by the approximate spanning tree algorithm is not an exact MST. If the AMST obtained is not an exact MST then some ϵ-smallest edge that was found between a tree T and $(S-T)$ was not the smallest edge between T and $(S-T)$. Let us call such an ϵ-smallest edge an *incorrect* edge. Let (p_i,p_j) be an incorrect edge, where $p_i \in T$ and $p_j \in (S-T)$; $r = d_q(p_i,p_j)$, where L_q is the distance metric under consideration; $r' = r/(1+2\epsilon)$; and $r'' = (1+2\epsilon)r$. Let $m = \left\lfloor \epsilon^{-(k-1)}(\log n)^{k-1} \right\rfloor$. Then there is no point in $(S-T)$ in the interior of the r'-neighborhood of p_i and there is no point in T in the interior of the r'-neighborhood of p_j. The intersection of the r'-neighborhoods of p_i and p_j defines a forbidden region in which no point in S may be located. The volume of this forbidden region is c_1 times the volume of the r'-neighborhood of p_i, where c_1 is some constant dependent on the dimension k, the distance metric L_q and ϵ. In addition, the shell around p_i, of radius r' and thickness $(r''-r')$ (i.e. the set of points whose L_q distance from p_i is at least r' and at most r''), contains at least m points.

Let P_a be the probability that the AMST has an incorrect edge. Let $e(p_i,p_j)$ be the event that (p_i,p_j) is an incorrect edge in the AMST and that p_i was in some tree T and p_j was in $(S-T)$, when (p_i,p_j) was obtained by the algorithm. Then

$$P_a \leq n^2 \Pr(e(p_i,p_j)).$$

We have

$$\Pr(e(p_i,p_j)) = \int \Pr(length((p_i,p_j)) = r)\ P(r)\ dr$$

where $P(r) = \Pr(e(p_i,p_j) / length((p_i,p_j)) = r)$. This gives

$$\Pr(e(p_i,p_j)) \leq \max_r P(r).$$

Let $P_S(r)$ be the probability that a shell around p_i of radius r' and thickness $(r''-r')$ contains at least m points and $P_1(r)$ the probability that a particular point lies in the shell. Then we have

$$P_S(r) \leq \binom{n-2}{m} (P_1(r))^m$$

Let $P_F(r)$ be the probability that the forbidden region defined by (p_i,p_j) does not contain any point in S and $P_2(r)$ the probability that a particular point lies in the for-

bidden region. Then

$$P_F(r) = (1 - P_2(r))^{n-2}.$$

Since $P(r) \leq P_S(r)$ and $P(r) \leq P_F(r)$, we get

$$\max_r P(r) = \max\{\max_{r<r_0} P_S(r), \max_{r\geq r_0} P_F(r)\}.$$

Choose r_0 such that $P_1(r_0) = \frac{(\epsilon^{-1}\log n)^{k-2}}{n}$; and ϵ such that $P_2(r) \geq \frac{c_2 P_1(r)}{\epsilon^{-(k-2)}(\log n)^{k-3}}$, where $c_2 \geq 3$. Then, for $k \geq 3$, we have

$$\max_{r<r_0} P_S(r) \leq P_S(r_0) \leq \frac{1}{n^{\log n \log\log n}}$$

and

$$\max_{r\geq r_0} P_F(r) \leq P_F(r_0) \approx \frac{1}{n^{c_2}}.$$

This gives $\Pr(e(p_i,p_j)) \leq 1/(n^{c_2})$ and $P_a = o(1/n)$.

6. Conclusion

We have developed a fast heuristic for finding approximate minimum spanning trees on the complete graph on n points in E_q^k, for the L_q, $q=2,3,...$, distance metrics. The weight of the AMST produced by the algorithm is at most $(1+\epsilon)$ times the weight of an MST. Moreover, if the n input points are assumed to be independently and uniformly distributed in the box $[0,1]^k$, then the probability that the AMST produced by the algorithm is an exact MST is $(1 - o(1/n))$.

Acknowledgements

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

References

[1] J. L. Bentley and H. A. Maurer, Efficient worst-case data structure for range searching, *Acta Informatica*, 13, 1980, pp. 155-168.

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

[3] K. Clarkson, Fast Expected-Time and Approximation algorithms for Geometric Minimum Spanning Trees, *Proc.* 16^{th} *Annual Symp. Theory of Comput.*, 1984, pp. 342-348.

[4] D. Cheriton and R. E. Tarjan, Finding minimum spanning trees, *SIAM J. Comput.*, 5, 1976, pp.724-742.

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

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

[7] H. Gabow, J. Bentley, and R. Tarjan, Scaling and Related Techniques for Geometric Problems, *Proc.* 16^{th} *Annual Symp. Theory of Comput.*, 1984, pp. 135-143.

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

[9] G. S. Lueker, A data structure for orthogonal range queries, *Proc.* 19^{th} *Annual IEEE Symp. Found. of Comp. Sci.*, 1978, pp. 28-34.

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

[11] E. M. Reingold, J. Nievergelt and N. Deo, *Combinatorial Algorithms: Theory* and *Practice*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

[12] M. I. Shamos and D. J. Hoey, Closest-point problems, *Proc.* 16^{th} *Annual Symp. Found. Comp. Sci.*, 1975, pp. 151-162.

[13] D. E. Willard, Predicate-oriented database search algorithms, Harvard University, Cambridge, MA, Aiken Computer Lab., Ph.d. Thesis, 1978, Report TR-20-78.

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

[15] A. C. Yao, On constructing minimum spanning trees in k-dimensional spaces and related problems, *SIAM J. Comput.*, 11, 1982, pp. 721-736.

[16] C. T. Zahn, Graph-theoretical method for detecting gestalt clusters, *IEEE Trans. Comput.*, C-20, 1970, pp. 68-70