Faster Geometric k-point MST Approximation

David Eppstein[*]

Department of Information and Computer Science
University of California, Irvine, CA 92717

Tech. Report 95-13

March 17, 1995

Abstract

We give fast new approximation algorithms for the problem of choosing k planar points out of n to minimize the length of their minimum spanning tree (equivalently, of their traveling salesman tour or Steiner tree). For any $x \leq k$, we can find an approximation achieving approximation ratio $O(\log k/\log x)$ in time $O(n \log n + 2^x kn \log k)$. In particular, we get an approximation with ratio $O(\log k/\log\log n)$ in time $O(kn^{1+\epsilon})$.

[*]Work supported in part by NSF grant CCR-9258355 and by matching funds from Xerox Corp.

1 Introduction

In this paper we consider the *k-minimum spanning tree* problem: given n points in the Euclidean plane, find the shortest tree spanning k of the points.

Up to constant factors in the approximation ratio, the k-MST problem is be equivalent to asking for a path connecting k points (the k-TSP problem) or a Steiner tree connecting k points. The choice of Euclidean metric is also not critical. However we will continue to use the k-MST formulation for simplicity. The k-MST problem was introduced independently by Zelikovsky and Lozevanu [9], and by Ravi et al. [8]. Ravi et al. described an approximation algorithm with approximation ratio $O(k^{1/4})$; this was quickly improved to $O(\log k)$ by Garg and Hochbaum [7], and to $O(1)$ by Blum et al. [3]. Many similar k-point selection problems with other optimization criteria can be solved in polynomial time [5, 6] but the k-MST problem is NP-complete [8, 9] (as are obviously the k-TSP and k-Steiner tree variants). Zelikovsky and Lozevanu [9] also consider a dual problem in which, given a length bound, one wishes to find the largest set of points with MST length within the bound; this formulation has different approximation behavior than the k-MST problem and may have a constant factor approximation (Zelikovsky, personal communication). For related work on non-geometric k-MST problems see [1, 4, 8, 9].

Although the $O(\log k)$ approximation ratio of the Garg-Hochbaum algorithm is fairly good, that paper did not spend as much attention on running time; an analysis of their technique shows that it takes total time $O(n^2k^4 + n^3)$. The very recent $O(1)$ approximation of Blum et al. is based on a complicated dynamic program which considers all possible axis-aligned rectangles determined by four points; its running time seems to be $k^{O(1)}n^5$.

In this paper we show how to improve the running times of both of these algorithms We first describe a modification of the Garg-Hochbaum planar k-MST approximation algorithm which achieves the same $O(\log k)$ approximation ratio but which improves the running time from the $O(n^2k^4)$ bound of [7] to $O(n \log n + nk \log k)$. For simplicity of exposition, we assume an integer model of computation, so that the "quadtree framework" construction of [2] is available.

We next describe a modification of this algorithm which can be used to achieve a better approximation ratio, at the expense of slightly increased runtime. For any $x \leq k$, we can find an approximation achieving approximation ratio $O(\log k/\log x)$ in time $O(n \log n + 2^x kn \log k)$. Choosing $x = (\log n)^\epsilon$ results in a polynomial time approximation algorithm with ratio

$O(\log k/\log\log n)$.

We then show how to achieve the same approximation ratio and time bounds in the Real RAM model of computation more commonly used in computational geometry, by rounding the input points to an appropriately chosen integer grid.

Finally, we show how the general neighborhood searching techniques of [6] and [5] can be used to implement a version of Blum et al.'s $O(1)$-approximation algorithm in time $k^{O(1)}n + O(n\log n)$. These techniques can be used to speed up any exact or approximate algorithm for k-MST approximation; in particular we can also solve the problem exactly in time $2^{O(k\log k)}n + O(n\log n)$.

2 Faster $O(\log k)$-approximation

To achieve an approximation ratio of $O(\log k)$, Garg and Hochbaum approximate the length of a spanning tree using a potential function defined in terms of a sequence of square grids. They then show that the subset of k points minimizing this potential function can be found using a form of dynamic programming. Their approximate k-MST is found as the MST of this subset.

Let s be an axis-aligned square in the plane; define the *size* $|s|$ to be the length of a side of s. We define an infinite sequence of grids $G_i(s)$ as follows: $G_0(s)$ is just the single square s itself, and grid $G_i(s)$ is formed by subdividing each square in grid $G_{i-1}(s)$ into four smaller squares. This construction is essentially just an infinite quadtree.

Given a set X of points, define the *potential function* $P_s(X)$ to be $\sum_{i=0}^{\infty} n_i|s|/2^i$, where n_i denotes the number of squares of $G_i(s)$ containing points in X. In other words, we sum $|s'|$ over all grid squares s' of different sizes that contain points of X. As Garg and Hochbaum show, this potential function is logarithmically related to the length of the minimum spanning tree of X:

Lemma 1 *Let T be a tree with total length ℓ, let X be the set of n vertices of T, and let s be a square containing T, with $|s| = O(\ell)$. Then $P_s(X) = O(|T|\log n)$.*

Proof: The contribution to $P_s(X)$ from G_i is at most $n|s|/2^i$, so values of i greater than $\log n$ contribute a total of $2|s| = O(\ell)$. The contribution from G_i for which $n_i \le 8$ also adds in a geometric series to $O(|s|) = O(\ell)$.

For each of the remaining values of i, choose a set of $\Theta(n_i)$ squares in G_i, no two adjacent, containing points of X. Since $n_i > 8$, there are at least two chosen squares. Since T must connect each of the chosen squares, there must be a path in T from any chosen square s' to some other chosen square, of length at least $|s'| = |s|/2^i$. Since there are $\Theta(n_i)$ chosen squares, and each path can be counted at most twice, the total length of all these paths and hence of T is $\Omega(n_i|s|/2^i)$. Summing this bound over the $O(\log n)$ remaining values of i gives the result. $\square$

Lemma 2 *Let X be a set of points contained in square s. Then there is a tree T spanning s with length $\ell = O(P_s(X))$.*

Proof: We connect each point of X with the center of the largest grid square containing only that point, and connect the center of each square with the center of the square twice the size that contains it. The total length involved in connections to squares in grid G_i is thus $O(n_i|s|/(2^i))$, and adding over all values of i gives the result. $\square$

Our approximation algorithm works by finding a collection of squares such that for any tree T, some square s contains T and satisfies $|s| = O(\ell(T))$. We select the square s and k-element subset $X \subset s$ minimizing $P_s(X)$, and output the minimum spanning tree of X.

The first step of the algorithm consists of selecting the collection of squares in which to optimize $P_s(X)$. We normalize our point set to lie within the unit square. We then start with three squares s_0, s_1, and s_2, each having size three with square s_j having point $(-j, -j)$ as its lower left corner. Our collection initially consists of all squares in each grid $G_i(s_j)$.

Lemma 3 *Let T be any tree with length ℓ, contained in the unit square. Then for some i and j, some square $s \in G_i(s_j)$ contains T and has size $|s| = O(\ell)$.*

Proof: Note that the initial squares s_j form a triple of grids in which each grid is horizontally and vertically offset by 1/3 of the grid size from each other grid. If we subdivide such a triple of grids, the same property remains true.

Choose i so that the squares in G_i have size between 3ℓ and 6ℓ. Then a simple case analysis shows that each point in the plane is at distance at least $\ell/2$ from the square boundaries in one of the three grids s_j. In particular this is true for the center of a bounding box of T, and T can extend for distance at most $\ell/2$ from this center point, so T is contained in a square of $G_i(s_j)$. $\square$

To achieve a polynomial algorithm, we need to reduce the number of squares in our collection to a polynomial. We need only optimize $P_s(X)$ among such squares s for which no smaller square s' in the same sequence of grids contains the same set of points; for otherwise $P'_s(X) < P_s(X)$ and we would output a tree in s' in preference to the one in s. As we now show, there are only $O(n)$ such squares.

The following lemma is proved in [2] (the results in that paper are phrased in terms of quadtrees; note that each of the squares we want is part of a finite quadtree defined by the input points). Note that for any pair of squares taken from grids $G_i(s)$, either one square contains the other, or both are disjoint; therefore the containment relations among a collection of squares from $G_i(s)$ can be represented as a tree.

Lemma 4 *In any sequence of grids $G_i(s)$ there are at most $n-1$ grid squares for which no smaller grid square contains the same set of points, and the set of such squares together with a tree representing its containment relations can be constructed in time $O(n \log n)$.*

Proof: The result follows by sorting the points to the order they would be examined in a traversal of their quadtree. Comparisons with respect to this order can be performed by examining the binary representations of the points' coordinates, so this sorting step can be done in $O(n \log n)$. The squares we are interested can be found as the minimal grid squares containing pairs of points adjacent in the sorted order. For details see [2]. $\square$

Define $m_i(s)$ to be the minimum potential $P_s(X)$ among i-point subsets X of the points in square s. We are interested in $m_k(s)$ but we will need to compute $m_i(s)$ for other values $i < k$ as well.

If only one of the four grid squares composing s contains any input points, let s' be the smallest grid square containing all these points; $m_i(s)$ can be computed in constant time using the formula

$$m_i(s) = m_i(s') + 2|s| - 2|s'|.$$

If two of the four squares contain input points, let a and b be those two squares. Then $m_i(s)$ can be computed in time $O(k)$ using the formula

$$m_i(s) = |s| + \min_{0 \le j \le i} m_j(a) + m_{i-j}(b).$$

And if more than two of the four squares contain input points, their values of m_j can be merged in pairs with a similar formula, again taking time $O(k^2)$. Thus all values $m_i(s)$, $0 \leq i \leq k$, for all the $O(n)$ squares selected in Lemma 4 can be computed in total time $O(nk^2)$.

We next analyze this computation more carefully to improve the running time. In many of the squares, one or more of the smaller squares s' composing it has fewer than k input points; in that case we know that $m_j(s') = 0$ for large values of j, and we only need examine the smaller values when computing $m_i(s)$. In general, the time to compute a value of $m_i(s)$ is at most proportional to the second largest value among the numbers of points in each of the four smaller squares.

Because of the tree structure of the grid squares, for any x, there are $O(n/x)$ grid squares in which this second largest value is x or more, so the total time to compute all values of $m_i(s)$ in all grid squares is $O(nk \log k)$.

Putting these ideas together, we have the following approximation result:

Theorem 1 *The k-MST problem can be approximated within a factor of $O(\log k)$ in time $O(n \log n + nk \log k)$.*

Proof: We find the sets of $O(n)$ grid squares as above, in time $O(n \log n)$. We then calculate $m_k(s)$ in each square, using also the other values of $m_i(s)$ in the grid squares, in total time $O(nk \log k)$. We choose our approximation to be the MST of the set X of points giving the minimum value of $m_k(s)$ over all the squares. If we let T be the true optimum, and s' be the smallest grid square containing T, then by Lemma 1 $m_k(s) \leq m_k(s') = O(\ell(T) \log k)$. Then by Lemma 2, $\ell(MST(X)) = O(m_k(s)) = O(\ell(T) \log k)$. $\square$

3 Better Approximation Factors

We now show how to achieve better approximation ratios than those of Theorem 1. The main idea is the following. The previous logarithmic approximation ratio came from the fact that in the definition of the potential function $P_s(X)$, we use a sequence of square grids with the square size in each grid half that of the previous one, so that $O(\log k)$ grids are required before the squares are small enough that their contribution to the potential is negligable. Instead, we use a sequence of grids with the square size in each grid only $1/r$ that of the previous one, for some $r > 2$; therefore $O(\log_r k)$ grids will contribute to the potential function and we will achieve an approximation factor of $O(\log_r k)$. However, the definition of the potential function is more complicated and we will need time exponential in r to find the point set with minimum potential. We now make this idea more explicit.

As before, let s be an axis-aligned square in the plane; let r be a positive integer. We define an infinite sequence of grids $G_{i,r}(s)$ as follows: $G_{0,r}(s)$ is just the single square s itself, and grid $G_{i,r}(s)$ is formed by subdividing each square in grid $G_{i-1,r}(s)$ into r^2 smaller squares of size $|s|/r^i$.

Given a set X of points, and a grid square s', define a point set C as follows: find the r^2 squares into which s' is divided, select the subset of squares containing points of X, and let C be the centers of the squares in that subset. Define $\tau(s', X)$ to be the length of the minimum spanning tree of C.

Now define the *potential function* $P_{s,r}(X)$ to be $\sum_{s'} |s'| + \tau(s', X)$, where the sum is taken over all squares in grids $G_i(s)$ that contain points in X.

Lemma 5 *Let T be a tree with total length ℓ, let X be the set of n vertices of T, and let s be a square containing T, with $|s| = O(\ell)$. Then $P_{s,r}(X) = O(|T| \log_r n)$.*

Proof: Define n_i to be the number of nonempty squares at level i. The contribution to $P_s(X)$ from G_i is at most $n|s|/r^i$, so values of i greater than $\log_r n$ contribute a total of $2|s| = O(\ell)$. The contribution from G_i for which $n_i \le 8$ also adds in a geometric series to $O(|s|) = O(\ell)$.

For each of the remaining values of i. choose a set of $\Theta(n_i)$ squares in G_i, no two adjacent, containing points of X. Since $n_i > 8$, there are at least two chosen squares. Since T must connect each of the chosen squares, there must be a path in T from any chosen square s' to some other chosen square, of length at least $|s'| = |s|/2^i$. Since there are $\Theta(n_i)$ chosen squares, and each path can be counted at most twice, the total length of all these paths and hence of T is $\Omega(n_i|s|/2^i)$. Summing this bound over the $O(\log_r n)$ remaining values of i accounts for the $|s'|$ terms in $P_{s,r}(X)$.

Within each square s', T must form paths in X connecting the small squares having centerpoints in C with themselves and with the boundary of s'. These paths, together with the boundary of s' and some additional short segments within the small squares, form a Steiner tree of C which therefore has length $\Omega(\tau(s', X))$. Thus the portion of T within s' has length $\Omega(\tau(s', X)) - O(|s'| + \sum |s''|)$ where the sum is taken over the smaller squares in s'. Summing this bound over the $O(\log_r n)$ sizes of squares accounts for the $\tau(s', X)$ terms in $P_{s,r}(X)$. $\square$

Lemma 6 *Let X be a set of points contained in square s. Then there is a tree T with length $\ell = O(P_{s,r}(X))$.*

Proof: We connect each point of X with the center of the largest grid square containing only that point. Within each square s' containing multiple points, we form the set C described above and connect the points of C and the center of s' itself in a single minimum spanning tree. The total length involved in connections in square s' is thus $O(|s'| + \tau(s', X))$, and adding over all values of s' gives the result. $\Box$

Define $m_{i,r}(s)$ to be the minimum potential $P_{s,r}(X)$ among i-point subsets X of the points in square s. We are interested in $m_{k,r}(s)$ but we will need to compute $m_{i,r}(s)$ for other values $i < k$ as well.

Lemma 7 *Suppose we know all values $m_{i,r}(s')$ for the r^2 small squares into which s is subdivided, and there are x input points in the square s' with the second most points in it. Then we can compute all values $m_{i,r}(s)$ in time $O(2^{r^2} r^2 kx)$.*

Proof: We try separately each of the 2^{r^2} ways of choosing a subset of the small squares, and find the optimal set X for each subset. For each choice, the minimum spanning tree of the set C can be found in time $O(r^2)$ (or faster if we use a dynamic minimum spanning tree algorithm) after which we can combine the values $m_{i,r}(s')$ in $O(r^2)$ pairwise merges, each taking time $O(kx)$. $\Box$

As a consequence, for r a power of two we can compute all values $m_{i,r}(s)$ in all quadtree squares in time $O(2^{r^2} r^2 (\log r) kn \log k)$. The factor of $\log r$ arises since the quadtree is effectively partitioned into $O(\log r)$ bushy trees by grouping levels according to their values modulo $\log_2 r$.

Theorem 2 *For any x, the k-MST problem can be approximated within a factor of $O(\log_x k)$ in time $O(n \log n + 2^x nk \log k)$.*

Proof: As before we form three grids $G_i(s_j)$ in which each square is subdivided into four half-size squares. We then choose r to be the largest power of two satisfying $2^{r^2} r^2 \log r \leq 2^x$. Each of the r^2 squares into which a square of $G_i(s_j)$ is subdivided will itself be a grid square in $G_{i+\log_2 r}(s_j)$. We find a collection of $O(n)$ grid squares, one of which covers T, as in the previous section in time $O(n \log n)$. We then calculate $m_{i,r}(s)$ in each grid square, using also the other values of $m_{i,r}(s)$ in r^2 smaller grid squares, in total time $O(2^x nk \log k)$.

We choose our overall approximation to be the MST of the set X of points giving the minimum value of $m_k(s)$ over all the grid squares. If we let T be the true optimum, and s' be the smallest grid square containing T, then by Lemma 5 $m_{k,r}(s) \leq m_{k,r}(s') = O(\ell(T) \log_r k) = O(\ell(T) \log_x k)$. Then by Lemma 6, $\ell(MST(X)) = O(m_{k,r}(s)) = O(\ell(T) \log_x k)$. $\square$

Corollary 1 *For any $\epsilon > 0$, the k-MST problem can be approximated within a factor of $O(\log k / \log \log n)$ in polynomial time $O(n \log n + 2^{(\log n)^\epsilon} nk \log k)$.*

4 Handling Real-number Input Coordinates

The algorithms above, based as they are on the quadtree framework of [2], require the input points to be represented with integer coordinates in which bitwise operations such as exclusive or may be performed in constant time.

In the Real RAM model more commonly used in computational geometry, input points have real-valued coordinates. The operations allowed consist of basic arithmetic such as addition and multiplication (often together with the extraction of roots of bounded-degree polynomials), but the floor and ceiling operations needed to convert coordinates to integers are forbidden. However, in order to permit array indexing, integer operations are allowed on integers of $O(\log n)$ bits or fewer.

We now describe how to implement our approximation algorithms in the same time bounds as above, on the Real RAM model. Our approach is to translate the input coordinates into $O(\log n)$-bit integers, such that the optimal k-MST length does not change by more than a constant, using operations permitted in the Real RAM model. After this translation, we run our integer algorithm as before.

We first compute a rough estimate of the k-MST length, so that we can know how much we can distort our input configuration while preserving the optimum k-MST. To do this we use the following result.

Lemma 8 (Eppstein and Erickson [6]) *In $O(n \log n + kn)$ we can find for each input point the smallest square centered on that point and containing exactly k input points.*

We let δ denote the side length of the smallest such square; then there is a set of k points with a tree of length $O(k\delta)$, and conversely any k-MST has length $\Omega(\delta)$. Therefore δ approximates the k-MST within an $O(k)$ factor. If we distort the input by moving every point by a distance of $\epsilon = c\delta/k$, we increase or decrease the length of any edge in the k-MST by at most $2c\delta/k$, so the total length change is $O(c\delta)$. By choosing c sufficiently small, such a distortion will change the k-MST length by a factor arbitrarily close to one. On the other hand, if any two points are at distance $\Omega(k\delta) = f$ from each other, they can not both be part of the optimal k-MST, and we can distort the distance between them arbitrarily as long as we keep them at distance at least f.

We use a construction similar to the "degraded grid" of Datta et al. [5]. We treat the x- and $y-$ coordinates independently; in each case we convert the coordinates to integers that are $O(nk^2) = O(n^3)$, so that $O(\log n)$ bits are required to represent them. A distance of one in the integer coordinates will correspond to a distance of ϵ in the original point set, where ϵ is chosen as above.

To convert the coordinates to integers, we first sort them. We partition the sorted list into *groups*, where each group is separated by a coordinate difference of f or more; we process the groups in order from smaller coordinates to larger ones. The *leader* of a group is the smallest coordinate of any point in the group. To begin the processing, the leader of the first group is assigned the integer zero. Suppose this leader corresponded to an input point with coordinate value a. Then any point in the group with coordinate value b is assigned the integer value closest to $(b-a)/\epsilon$. After each group is assigned integers in a range $[i, j]$, we begin the next group by assigning its leader the next integer larger than $j + f/\epsilon$, where $f = \Omega(k\delta)$ is defined as above.

Lemma 9 *If we perform the integer assignment process described above, and scale the resulting integer grid by a factor of ϵ, we get a point set in which each distance less than f is distorted by a total amount of $O(\epsilon)$, and each distance greater than $f + \Omega(\epsilon)$ is distorted to a new distance greater than f.*

Proof: If two points are at distance $O(f)$ from each other, their x- and y-coordinates must both belong to the same groups. If we form a point by combining the x- and y-coordinate group leaders, and translate the scaled integer grid so that this point coincides with the corresponding integer grid point, then by construction the two given points are at distances of $O(\epsilon)$ from their corresponding integer points; therefore their distance is distorted by $O(\epsilon)$.

Conversely, if two points belong to different groups, they are separated by an empty strip of width f or more and are therefore at distance at least f. The assignment of leader values causes this distance bound to be preserved. $\Box$

Theorem 3 *In time $O(n \log n + kn)$ in the Real RAM model, we can translate the input points to points on an $O(nk^2)$-value integer grid, such that the k-MST of the integer grid points provides an approximation with ratio arbitrarily close to one of the original k-MST.*

Proof: As described above, we find δ in time $O(n \log n + kn)$, and use this value to assign integer values to the coordinates; the remaining time bottleneck is sorting the coordinates, which takes time $O(n \log n)$. In the resulting sequences of x- and y-coordinates, successive points are at most $O(k^2)$ integer places apart, so the largest integer produced is $O(nk^2)$.

Since the optimal k-MST has no points at distances larger than f, we know that all distances within the k-MST are closely approximated by distances in its integer translation. The same is true for any other k-point spanning trees for which all points are within the same groups of x- and y-coordinates. A tree spanning two or more groups must contain an edge larger than f, and therefore be larger than the optimal k-MST. Therefore the optimal k-MST in the integer point set is a close approximation of the original k-MST. $\Box$

Corollary 2 *For any $\epsilon > 0$, the k-MST problem can be approximated on the Real RAM model of computation within a factor of $O(\log k / \log \log n)$ in polynomial time $O(n \log n + 2^{(\log n)^\epsilon} nk \log k)$.*

5 Faster constant-ratio approximation

We now show how to speed up any exact or approximate algorithm for geometric k-MST approximation, using ideas from [6] and [5]. Our results are based on the following fact.

Lemma 10 *Let T be the optimal k-MST in a set S of points, and let x be any vertex in T. Then the vertices of T are a subset of the $O(k^2)$ points in S nearest to x.*

Proof: Let ℓ denote the length of the k-MST. For some value c to be chosen, suppose there are at least ck^2 points in a disk of radius ℓ centered around x. Cover this area by ck squares, each of side length $2\ell/\sqrt{ck}$. Then some square s contains at least k points. Any minimum spanning tree of any k points within this square has length $O(\ell/c)$, and by choosing c to be an appropriate constant we can make this bound smaller than ℓ. $\square$

The basic idea is to find the $O(k^2)$ nearest neighbors to each input point, and find a k-MST approximation within each such set of neighbors. By the lemma above, the true k-MST is in one of these nearest neighbor sets, so this subdivision of the problem preserves approximation quality. Eppstein and Erickson [6] use a more complicated method to collect these sets of neighbors into $O(n/k^2)$ sets of $O(k^2)$ neighbors each.

Datta et al. [5] further modify this procedure to use a degraded grid instead of explicitly computing nearest neighbors, again for any problem in which a result like Lemma 10 holds. For any x they show how to find a collection of $O(n/x)$ sets S_i, each having $|S_i| = O(x)$, and such that if ℓ is the minimum radius of a set of x points then all sets of radius ℓ are contained in some S_i. As a result, we can speed up any k-MST algorithm as follows.

Lemma 11 *If we have any time bound $T(n,k)$ for an exact or approximate geometric k-MST problem, we can solve the same problem in time $O(n \log n + nT(k^2,k)/k^2)$.*

Proof: We use the method of Datta et al. with $x = ck^2$, for the constant c used in the proof of Lemma 10. This gives a collection of $O(n/k^2)$ sets S_i, each with $O(k^2)$ points, and such that the exact k-MST is contained in one of the S_i. We then apply the given exact or approximate algorithm within each S_i. $\square$

We apply this idea to Blum's constant factor approximation, and to the trivial $O(n^k k \log k)$ exhaustive search algorithm for exactly solving the k-MST problem. The algorithms we previously described in this paper are already fast enough that this idea does not give a further improvement.

Corollary 3 *The k-MST problem can be approximated within a factor of $O(1)$ in time $k^{O(1)}n + O(n \log n)$. The exact k-MST can be constructed in time $2^{O(k \log k)}n + O(n \log n)$.*

References

[1] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala. Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. In *Proc. 27th ACM Symp. Theory of Computing*, pages 277–283, 1995.

[2] M. Bern, D. Eppstein, and S.-H. Teng. Parallel construction of quadtrees and quality triangulations. In *Proc. 3rd Workshop on Algorithms and Data Structures*, pages 188–199. Springer-Verlag, LNCS 709, 1993.

[3] A. Blum, P. Chalasani, and S. Vempala. A constant-factor approximation for the k-MST problem in the plane. In *Proc. 27th ACM Symp. Theory of Computing*, pages 294–302, 1995.

[4] S. Y. Cheung and A. Kumar. Efficient quorumcast routing algorithms. In *Proc. IEEE INFOCOM '94 Conference on Computer Communications*, volume 2, pages 840–847, 1994.

[5] A. Datta, H.-P. Lenhof, C. Schwarz, and M. Smid. Static and dynamic algorithms for k-point clustering problems. In *Proc. 3rd Worksh. Algorithms and Data Structures*, pages 265–276. Springer-Verlag, LNCS 709, 1993.

[6] D. Eppstein and J. Erickson. Iterated nearest neighbors and finding minimal polytopes. *Discrete & Comput. Geom.*, 11:321–350, 1994.

[7] N. Garg and D. S. Hochbaum. An $O(\log k)$ approximation for the k minimum spanning tree problem in the plane. In *Proc. 26th ACM Symp. Theory of Computing*, pages 432–438, 1994.

[8] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi. Spanning trees short and small. In *Proc. 5th ACM-SIAM Symp. Discrete Algorithms*, pages 546–555, 1994.

[9] A. A. Zelikovsky and D. D. Lozevanu. Minimal and bounded trees. In *Tezele Cong. XVIII Acad. Romano-Americane, Kishinev*, pages 25–26, 1993.