Estimating the Weight of Metric Minimum Spanning Trees in Sublinear-Time*

Artur Czumaj
Department of Computer Science
New Jersey Institute of Technology
czumaj@cis.njit.edu

Christian Sohler
Heinz Nixdorf Institute and
Institute for Computer Science
University of Paderborn
csohler@uni-paderborn.de

November 8, 2003

Abstract

In this paper we present a sublinear time $(1+\epsilon)$-approximation randomized algorithm to estimate the *weight* of the minimum spanning tree of an n-point metric space. The running time of the algorithm is $\widetilde{\mathcal{O}}(n/\epsilon^{\mathcal{O}(1)})$. Since the full description of an n-point metric space is of size $\Theta(n^2)$, the complexity of our algorithm is *sublinear* with respect to the input size. Our algorithm is almost optimal as it is not possible to approximate in $o(n)$ time the weight of the minimum spanning tree to within any factor. Furthermore, it has been previously shown that no $o(n^2)$ algorithm exists that *returns a spanning tree* whose weight is within a constant times the optimum.

*Research supported in part by NSF grants CCR-0313219 and CCR-0105701, DFG grant Me 872/8-2, and IST program of the EU under contract IST-1999-14186 (ALCOM-FT).

1 Introduction

In this paper we consider the classical minimum spanning tree problem. Despite extensive investigations over last few decades, the complexity of the minimum spanning tree problem is not completely understood. Although an optimal deterministic algorithm is known [16], no tight bounds on the running time of this algorithm could be obtained. The best upper bound on the running time for a deterministic algorithm was obtained by Chazelle [4] who presented an algorithm that achieves a running time of $\mathcal{O}(|V| + |E|\,\alpha(|E|, |V|))$, where α is the functional inverse of Ackermann's function. In turns, Karger et al. [14] gave an optimal $\mathcal{O}(|V| + |E|)$-time randomized algorithm. A vast of research has been devoted to study the minimum spanning problem for various classes of graph and for variants of the problem. For example, the problem of computing the minimum spanning tree of a set of points in a Euclidean space and related problems have been intensively studied (see [8] for a summary of results). Despite this effort, the fastest algorithm to compute such a minimum spanning tree in the $\mathbb{R}^d$ requires $O(n^{2-2/(\lceil d/2\rceil+1)+\epsilon})$ for an arbitrary small constant ϵ.

In this paper we present another important step towards understanding the minimum spanning tree problem. We consider the classical variant of the *minimum spanning tree problem for metric spaces*, or equivalently, in graphs with *weights satisfying the triangle inequality*. The input to the problem consists of an n-point metric space (P, d) and the goal is to estimate the *weight* of the minimum spanning tree of P. In this paper, we show that even though the full description of an n-point metric space is of size $\Theta(n^2)$, there exists an algorithm that approximates the weight of the minimum spanning tree of P to within a $(1+\epsilon)$-factor in time $\widetilde{\mathcal{O}}(n/\epsilon^{\mathcal{O}(1)})$ (we use $\widetilde{\mathcal{O}}$ to hide poly-logarithmic factors). This is the first *sublinear-time* algorithm for this problem. Our algorithm is randomized and it achieves the promised approximation guarantee with the probability of at least $3/4$ (using standard techniques the probability can be amplified if needed). Furthermore, it was previously shown in [13] that no $o(n^2)$ algorithm exists that *returns a spanning tree* whose weight is within any constant times the weight of the minimum spanning tree of P. Therefore, our result shows that one can approximate the weight of the minimum spanning tree but there is no hope to find a *witness* for that approximation in sublinear time. Our running time is essentially asymptotically optimal, because it is easy to show that no $o(n)$-time algorithm exists that approximates the weight of the minimum spanning tree within *any* factor.

Our algorithm yields an interesting extension to the growing list of problems solvable in sublinear-time that are of large demand in massive data sets analysis. Due to the tremendous increase in computational power and interconnectivity during the last decade more and more often we have to deal with *massive* data sets, which are sets of size in the range of several Gigabytes or more. Examples for such massive data sets are Internet traffic logs, clickstream patterns, sales logs, and call-detail data records in telecommunication industry. Massive data sets typically cannot be processed by algorithms requiring more than linear time and often even linear time algorithms may be too slow. This leads to a natural question which problems of interest (if any) can be solved in *sublinear time*. Some simple results indicating that it is sometimes possible to solve certain approximation problems in sublinear time are well-known, for example approximating the median or the average value of a set of n numbers. More sophisticated results have been obtained in the last few years for a number of more complex problems, including clustering problems in metric spaces [1, 5, 12, 13, 15], graph problems [6, 9, 11], geometric problems [5, 7], matrix approximation [10], and edit distance approximation [2].

We notice also that by the well known relationship between minimum spanning trees, travelling salesman tours, and minimum Steiner trees (see, e.g., [17]), our algorithm for estimating the weight of the minimum spanning tree immediately yields *sublinear-time* $(2+\epsilon)$-approximation algorithms

for two other classical problems in metric spaces (or in graphs satisfying the triangle inequality): *estimating the weight of the travelling salesman tour* and *the minimum Steiner tree*. No $o(n^2)$-time algorithms have been known for these problems before. We believe that besides being interesting by themselves, these approximation results may find applications to bound the quality of solutions in subproblems used in branch and cut (bound) algorithms to compute the exact solution to these problems.

1.1 Related work

The problem of approximating the weight of the minimum spanning tree in sublinear time has been first studied for arbitrary graphs in *adjacency list* representation in a very recent paper by Chazelle et al. [6]. In [6], a *pseudo-sublinear* algorithm is given: if the maximum (or average) degree is D and if all edge weights are known to be in the interval $[1, W]$, then the algorithm approximates the weight of the minimum spanning tree to within an $(1+\epsilon)$ factor in time $\widetilde{\mathcal{O}}(D \cdot W \cdot \epsilon^{-3})$ with probability at least $3/4$.

If we apply their algorithm to the metric version of the problem then the running time is $\mathcal{O}(n \cdot W \cdot \epsilon^{-3})$ because $D = n-1$. Therefore, in our setting, their algorithm is sublinear only if the ratio between the longest and the shortest edge W is sublinear in n, what certainly does not have to be the case in general (for example, even in the case when (P, d) corresponds to the set of points P in a Euclidean plane, then it is known that W must be at least $\Omega(\sqrt{n})$ and hence the running time is $\Omega(n^{1.5} \cdot \epsilon^{-3})$).

In [7], the authors consider the problem of estimating the weight of the Euclidean minimum spanning tree of a set P of n points in the $\mathbb{R}^d$. In this paper it is assumed that the input point set is stored in a sophisticated data structure that supports two types of access operations, namely, (i) emptiness queries for axis parallel squares and (ii) approximate nearest neighbor queries for a set of prespecified cones. Additionally, it is assumed that a smallest axis parallel bounding box of P is given. In this model the authors give an algorithm that approximates the weight of the Euclidean minimum spanning tree of P within a relative error of ϵ. The algorithm has a running time of $\widetilde{\mathcal{O}}(\sqrt{n}/\epsilon^{\mathcal{O}(1)})$ assuming that the dimension is a constant and not counting the time for the access operations to the data structure. The algorithm uses $\widetilde{\mathcal{O}}(\sqrt{n}/\epsilon^{\mathcal{O}(1)})$ queries of type (i) and (ii).

1.2 New contribution

In contrast to both of the aforementioned algorithms, our algorithm does not make any assumption about the input besides the assumption that we can evaluate the distance between any two points in the metric space in constant time.

The high level approach of our algorithm is similar to that in [6]. We regard the metric space as a complete graph and we express the weight of its minimum spanning tree by a formula depending on the number of connected components in certain auxiliary subgraphs. For simplicity of presentation we assume that all distances are powers of $(1+\epsilon)$ and that the longest edge in (P, d) has length $W = 2n/\epsilon$ (as we will see later, these assumptions do not affect the complexity of the problem). We denote by $G^{(t)} = (P, E^{(t)})$ the graph that contains an edge between $p, q \in P$ if $d(p, q) \leq t$. We use a randomized procedure to approximate the number $c^{((1+\epsilon)^i)}$ of connected components in each subgraph $G^{((1+\epsilon)^i)}$. Using the identity $\text{MST} = n - W + \epsilon \cdot \sum_{i=0}^{\log_{1+\epsilon} W-1} (1+\epsilon)^i \cdot c^{((1+\epsilon)^i)}$ we obtain an estimator with expectation MST.

The advantage of our approach is that we have to estimate only $\log W$ times the number of the connected components of a certain threshold graph in contrast to W times in [6]. We achieve this at the cost of an increased variance of the estimator (which makes it impossible to apply this approach to arbitrary graphs considered in [6]). Instead of approximating the number of connected components within an additive error of ϵn as in [6], we obtain an approximation that either has a multiplicative error of $1+\epsilon$ or an additive error of $\epsilon \cdot \text{weight}(\text{MST})$. To achieve this result, we introduce an estimator that is based on a new graph traversal combined with a stochastic procedure to estimate the degree of a vertex in $G^{((1+\epsilon)^i)}$. Our graph traversal explores the triangle inequality to ensure trade-offs between the number of connected components, the vertex degrees, and the size of the minimum spanning tree, which can be used to show that our estimator is sharply concentrated around its expectation.

2 Preliminaries

We consider the problem of *estimating the weight of the minimum spanning tree in a metric space*: given access to the $n \times n$ distance matrix of a metric space (P, d), $|P| = n$, approximate the weight of the minimum spanning tree of P. Throughout the paper, we denote by MST the weight of the minimum spanning tree of P. Our main contribution is an algorithm that in $\widetilde{\mathcal{O}}(n/\epsilon^8)$ time computes a $(1+\epsilon)$-approximation of MST, i.e., outputs a value M such that $(1-\epsilon)\cdot \text{MST} \leq M \leq (1+\epsilon)\cdot \text{MST}$ with probability at least $3/4$. We will use ϵ as the approximation parameter throughout the paper.

Our result holds for arbitrary metrics (P, d) and assumes only that a constant-time access to the distance oracle is provided. However, for our analysis we will make two important assumptions that we justify now.

Our first assumption is that for every pair $p, q \in P$ we have $d(p,q) \in [1, 2n/\epsilon]$. Using standard transformation, such assumption may introduce an approximation error of at most $1+\epsilon$. Indeed, Indyk [13] showed that in $\mathcal{O}(n)$ time one can approximate to within factor 2 the longest distance in a metric space. Once we have such an approximation W, we can rescale the distances such that $W = 2n/\epsilon$. Since W is a 2-approximation of the largest distance, after the scaling all distances are in $[0, 2n/\epsilon]$. Furthermore, by the triangle inequality the weight of a minimum spanning tree of a metric space is at least as large as the weight of the longest distance and since the longest distance is at least $W/2$, we have $\text{MST} \geq n/\epsilon$. Next, we observe that rounding up every distance smaller than 1 to the distance 1 will change MST by an additive term of at most $n-1$ while preserving the triangle inequality. Since $n - 1 \leq \epsilon \cdot \text{MST}$, in the so modified metric the weight of a minimum spanning tree is an $(1+\epsilon)$-approximation of the weight of a minimum spanning tree in the original metric.

Furthermore, to simplify the presentation of our algorithm we assume that all distances are powers of $(1+\epsilon)$, that is, for every $p, q \in P$, we have $d(p,q) = (1+\epsilon)^i$ for certain integer $i \geq 0$. Intuitively, this assumption is justified by the fact that one can round up all distances to the nearest power of $(1+\epsilon)$ and such rounding changes MST by a factor of at most $(1+\epsilon)$. We notice however, that the rounding may invalidate the triangle inequality and therefore this assumption requires some more comments. At the end of the paper, in Section 7, we briefly discuss changes that are to be done in order to formally justify this assumption.

Therefore, from now on, unless stated otherwise, we will assume that all distances are powers of $(1+\epsilon)$ and lie in the interval $[1, 2n/\epsilon]$.

2.1 Approximating MST via counting connected components in auxiliary graphs

Our high level approach of approximating the weight of the minimum spanning tree is similar to the one used in [6]. We express the weight of the minimum spanning tree in terms of the number of connected components in certain auxiliary graphs. For a given threshold $t \in \mathbb{R}$ we say that two points $p, q \in P$ are t-*close*, if their mutual distance is at most t. We say p and q are in the same t-*connected component* if they are in the same equivalence class of the transitive closure of the "t-close" relation (that is, if there is a sequence of points $x_0, x_1, \ldots, x_\ell$ with $x_0 = p$ and $x_\ell = q$ such that x_i and x_{i+1} are t-close for all $0 \leq i < \ell$).

Let us denote by $c^{(t)}$ the number of t-connected components of (P, d). Then we can write:

$$\text{MST} \;=\; n - W + \epsilon \cdot \sum_{i=0}^{\log_{1+\epsilon} W - 1} (1+\epsilon)^i \cdot c^{((1+\epsilon)^i)} \;, \tag{1}$$

where $W = 2n/\epsilon$ denotes an upper bound for the distances in (P, d) and all distances are powers of $(1+\epsilon)$.

Our approach is to compute a randomized estimator $\hat{c}^{((1+\epsilon)^i)}$ for each $c^{((1+\epsilon)^i)}$. Using the estimator we can phrase now our randomized algorithm:

METRIC-MST-APPROXIMATION (P, ϵ)
  for $i = 0$ **to** $\log_{1+\epsilon} W - 1$ **do**
    Compute estimator $\hat{c}^{((1+\epsilon)^i)}$ for $c^{((1+\epsilon)^i)}$
  Output $M = n - W + \epsilon \cdot \sum_{i=0}^{\log_{1+\epsilon} W - 1} (1+\epsilon)^i \cdot \hat{c}^{((1+\epsilon)^i)}$

To analyze the performance of the Metric-MST-Approximation algorithm we introduce two parameters ζ and ρ. Parameter ζ measures the quality of estimating the value $c^{((1+\epsilon)^i)}$ and parameter ρ measures the error probability of the estimator. Our novel contribution is a sublinear-time randomized algorithm that outputs an estimator $\hat{c}^{((1+\epsilon)^i)}$ that with probability at least $1 - \rho$ satisfies the following property:

$$(1-\zeta) \cdot c^{((1+\epsilon)^i)} - \zeta \cdot \frac{\text{MST}}{\epsilon \cdot (1+\epsilon)^i} \;\leq\; \hat{c}^{((1+\epsilon)^i)} \;\leq\; (1+\zeta) \cdot c^{((1+\epsilon)^i)} + \zeta \cdot \frac{\text{MST}}{\epsilon \cdot (1+\epsilon)^i} \;. \tag{2}$$

For our algorithm we will set $\rho = \frac{1}{1+4\log_{1+\epsilon} W} = \Theta\left(\frac{\epsilon}{\ln(n/\epsilon)}\right)$ and $\zeta = \frac{\epsilon}{3+\log_{1+\epsilon} W} = \Theta\left(\frac{\epsilon^2}{\ln(n/\epsilon)}\right)$. This implies that with probability $(1 - \frac{1}{1+4\log_{1+\epsilon} W})^{\log_{1+\epsilon} W} > e^{-1/4} > 3/4$ all estimators $\hat{c}^{((1+\epsilon)^i)}$ satisfy inequality (2). By basic calculations, this yields the following inequalities that hold with probability at least $3/4$:

$$(1-\epsilon) \cdot \text{MST} \;\leq\; M \;\leq\; (1+\epsilon) \cdot \text{MST} \;.$$

In Sections 3 – 6 we describe details of our randomized algorithm that in $\widetilde{\mathcal{O}}(n \cdot \zeta^{-3} \cdot \rho^{-1} \cdot \epsilon^{-1})$ time computes the estimator $\hat{c}^{((1+\epsilon)^i)}$ that satisfies inequality (2). This will conclude the proof of our main theorem:

Theorem 1 *Let $0 < \epsilon < 1$ be an approximation parameter. Given access to the $n \times n$ distance matrix of a metric space (P, d), $|P| = n$, algorithm* METRIC-MST-APPROXIMATION *computes in $\widetilde{\mathcal{O}}(n/\epsilon^8)$ time a value M such that with probability $3/4$,*

$$(1-\epsilon) \cdot \text{MST} \;\leq\; M \;\leq\; (1+\epsilon) \cdot \text{MST} \;.$$

Let us observe that this result is almost optimal since it is easy to see that any constant-factor algorithm requires time $\Omega(n)$ even in randomized setting (see Theorem 3 in Section 8).

3 Estimating the number $c^{(t)}$ of t-connected components: Main ideas

In this section we show the main ideas of our algorithm for estimating the number $c^{(t)}$. We begin with the definition of the *threshold graph* $G^{(t)} = (P, E^{(t)})$ as the graph with vertex set P that contains an edge between $p, q \in P$ if and only if $d(p, q) \leq t$; in other words, $E^{(t)} = \{(p, q) : p, q \in P \text{ and } d(p, q) \leq t\}$. Notice that the connected components of $G^{(t)}$ are the t-connected components of (P, d).

In this section we assume that t is a power of $(1 + \epsilon)$ and present a high level description of a randomized process that outputs a value $\hat{c}^{(t)}$ which is an approximation of $c^{(t)}$ that satisfies inequality (2). Our randomized process repeats the following procedure until a certain threshold value is reached, to ensure that the estimation of the number of t-connected components is with high probability close to $c^{(t)}$.

- Pick a starting vertex $p \in P$ uniformly at random.
- Choose a random integer number X according to the probability distribution $\mathbf{Pr}[X \geq k] = 1/k$.
- Verify whether the connected component in $G^{(t)}$ containing vertex p has at most X vertices or it has more than X vertices.

With the exception of a minor modification in the probability distribution of X, the scheme above has been proposed by Chazelle et al. [6]. We will run this procedure multiple times and in each repetition of this procedure we output β_i that is the indicator random variable that in the ith trial the connected component has at most X vertices. That is, if we denote by $n_p^{(t)}$ *the size of the connected component in* $G^{(t)}$ *containing vertex* p, then $\beta_i = 1$ if $n_p^{(t)} \leq X$ and $\beta_i = 0$ otherwise. Notice that,

$$\mathbf{E}[\beta_i] \;=\; \sum_{\text{connected component } C \text{ in } G^{(t)}} \mathbf{Pr}[p \in C] \cdot \mathbf{Pr}[X \geq |C|] \;=\; \sum_{\text{connected component } C \text{ in } G^{(t)}} \frac{|C|}{n} \cdot \frac{1}{|C|} \;=\; \frac{c^{(t)}}{n} \;.$$

Therefore, if there are s repetitions of the procedure above then we define

$$\hat{c}^{(t)} \;=\; \frac{n}{s} \cdot \sum_{i=1}^{s} \beta_i \;.$$

Since by the arguments above $\mathbf{E}[\hat{c}^{(t)}] = c^{(t)}$, this motivates the use of $\hat{c}^{(t)}$ as an estimator of the number of connected components, see [6]. The challenging part required to complete the analysis is to show that the random variable $\hat{c}^{(t)}$ is sharply concentrated around its expectation and to show that it can be computed efficiently.

4 Towards approximating $c^{(t)}$ — The Clique-Tree Traversal

Our method to verify whether a given connected component in $G^{(t)}$ has the number of vertices smaller than or equal to certain threshold value X is to traverse the graph $G^{(t)}$ starting at vertex p. Chazelle et

al. [6] used the classical breadth-first search (BFS) traversal algorithm for this purpose. However, in our setting this algorithm is too slow and the corresponding random estimator has too large variance and therefore we have to develop a new traversal algorithm that is tuned to work well for metric graphs. The design of such an algorithm and its analysis are the main contributions of our paper.

Before we define our graph traversal we need a few more definitions. We call two vertices $p, q \in P$ *twins* in $G^{(t)}$, if they have the same neighborhood in $P \setminus \{p, q\}$. While performing a graph traversal the knowledge that a vertex q is a twin of another previously visited vertex p allows us to do not consider the outgoing edges of q in the graph traversal. Since both vertices are twins we know that every vertex reachable from q is also reachable from p. The following lemma provides a simple sufficient condition for two vertices to be twins.

Lemma 4.1 *Let* $t = (1+\epsilon)^i$ *for some* $i \in \mathbb{N}$ *and let* p, q *be two vertices with* $d(p,q) < \epsilon \cdot t$. *Then* p, q *are twins in* $G^{(t)}$.

Proof : By our assumption all distances in (P, d) are powers of $(1+\epsilon)$. Since $t = (1+\epsilon)^i$, there is no pair of points in P whose distance is larger than t but smaller than $(1+\epsilon)\,t$. If a vertex r is a neighbor of p in $G^{(t)}$, then by definition $d(p,r) \leq t$. Since $d(p,q) < \epsilon$, the triangle inequality implies that $d(q,r) < (1+\epsilon)\,t$ and hence, $d(q,r) \leq t$. This means that r is also a neighbor of q. By symmetry it follows that p and q are twins. □

We are now ready to explain our graph traversal in more details. At the beginning all vertices are *unexplored*. Then the starting vertex p is marked as *explored* and *representative*. In the next step all neighbors of p that are in distance less than $\epsilon\,t$ are marked as *explored*. Then we proceed similarly to Prim's algorithm for the computation of minimum spanning trees. Among all edges in $E^{(t)}$ that connect a representative vertex with an unexplored vertex we choose the shortest. This leads us to a new vertex that is again chosen to be *explored* and *representative*. Then, we repeat all steps above until the entire connected component in $G^{(t)}$ containing point p is explored. We call this graph traversal the *Clique-Tree-Traversal*.

We give a pseudocode for the Clique-Tree-Traversal below. The sets E, U, and R denote the sets of explored, unexplored, and representative vertices, respectively.

Clique-Tree-Traversal (P**,** p**,** t**,** ϵ**)**
$R = \{p\}$; $E = \{p\}$; $U = P \setminus \{p\}$
while there is an edge $e = (p,q) \in E^{(t)}$ with $p \in R$ and $q \in U$ **do**
  let (p,q) be the shortest such edge
  $E = E \cup \{q\}$; $U = U \setminus \{q\}$
  if $d(p,q) \geq \epsilon\,t$ **then** $R = R \cup \{q\}$

Let us now discuss some properties of the Clique-Tree-Traversal algorithm.

First, it is easy to see that the Clique-Tree-Traversal algorithm can be implemented to run in $\mathcal{O}(n \cdot |R|)$ time.

Next, we notice that the algorithm explores the entire connected component in $G^{(t)}$ in which the starting vertex p is located. This is because it considers all edges but the edges connecting to vertices whose twins have been previously visited. Therefore, in particular, at the end of the algorithm we have $|E| = n_p^{(t)}$.

The next important property of the Clique-Tree-Traversal algorithm is that it provides implicitly a lower bound on the size of the minimum spanning tree of that connected components. To see this, let us consider all edges that have been used in the algorithm to explore a new representative vertex. We call these edges *representative edges*. It is easy to see that the set of representative edges forms a tree. At the time when the new representative vertex is explored the corresponding representative edge is a shortest edge connecting the old representative vertices to the new representative vertex. Hence, it must be contained in a minimum spanning tree of the graph induced by the representative vertices (cf. Prim's algorithm). Since the minimum spanning tree of the representative vertices is a Steiner tree of the vertices of the connected component of p, it can have at most twice the weight of the minimum spanning tree of the whole component (cf. [17]). By the fact that every representative edge has length at least $\epsilon\, t$, $\epsilon\, t\,(|R|-1)/2$ is a lower bound on the size of the minimum spanning tree of the connected component of p.

We summarize our discussion in the following lemma.

Lemma 4.2 *The algorithm Clique-Tree-Traversal satisfies the following properties: (1) The algorithm can be implemented to run in time $\mathcal{O}(n \cdot |R|)$. (2) The explored vertices form exactly the connected component of p in $G^{(t)}$. (3) The representative edges form a minimum spanning tree of the graph induced by the representative vertices. (4) We have* $\text{MST} \geq \epsilon\, t\,(|R|-1)/2$. □

In the analysis of our algorithm we will also use the notion of *graph dispersion*. To define this notion, let us first extend the Clique-Tree-Traversal to a full graph traversal on $G^{(t)}$ in the following natural way: We start with an arbitrary vertex p and run the Clique-Tree-Traversal with parameters (P, p, t, ϵ). If not all vertices are explored at the end of this traversal, we start the Clique-Tree-Traversal from one of the unexplored vertices (we never start at the same connected component more than once). We do this until every vertex has been explored.

It is easy to see that the number of representative vertices computed by the full Clique-Tree-Traversal may depend on the starting vertices. One parameter of particular interest for our analysis is the so-called *dispersion* of the graph $G^{(t)}$, which is the maximum number of representative vertices $\mathcal{L}(G^{(t)})$ computed by the full Clique-Tree-Traversal for given P, t, and ϵ (the maximum is taken over all possible vertex orderings). We will use the dispersion of $G^{(t)}$ together with the property that all twins form a clique in $G^{(t)}$ to obtain bounds on the density of $G^{(t)}$. Furthermore, our main use of $\mathcal{L}(G^{(t)})$ is to obtain a lower bound for MST, as in the next lemma, which follows immediately from the definition of $\mathcal{L}(G^{(t)})$ and Lemma 4.2.

Lemma 4.3 *If $\mathcal{L}(G^{(t)}) > 1$, then $\mathcal{L}(G^{(t)}) \leq \text{MST}/(4 \cdot \epsilon \cdot t)$.* □

5 Estimating degrees of vertices: Degree-Estimate Algorithm

Let $\deg_t(p)$ denote the degree of vertex p in $G^{(t)}$. For our algorithm we need a procedure to estimate the degree of a given vertex p in $G^{(t)}$. Note that in our setting finding $\deg_t(p)$ *exactly* requires trivially $\Omega(n)$ time. However, we will need to *estimate* the degree with high accuracy in time inversely proportional to the degree.

Our use of estimating the degree is quite simple: in order to detect if the number $n_p^{(t)}$ of vertices in the connected component in $G^{(t)}$ containing p is smaller than or equal to a given integer X, we can first test if $\deg_t(p) < X$; if $\deg_t(p) \geq X$, then we certainly must have $n_p^{(t)} > X$.

The algorithm for estimating the degree of a given vertex follows a standard sampling approach.

Lemma 5.1 *Let (P, d) be a metric space with $|P| = n$ and let $G^{(t)}$ be the threshold graph for some real number t. Then, there exists an algorithm Degree-Estimate(P, t, p) that with probability at least $1 - 1/n^4$, runs in time $\mathcal{O}(n \log n / \deg_t(p))$, and returns value $\widehat{D}(p)$ such that $\frac{1}{2} \cdot \widehat{D}(p) \leq \deg_t(p) \leq 2 \cdot \widehat{D}(p)$.*

Proof : Let us first suppose that we know the value of $\deg_t(p)$. We choose (with replacement) at random $N = c \cdot n \cdot \log n / \deg_t(p)$ vertices for some large enough constant c. Let Γ be the random variable denoting the number of the chosen vertices that are adjacent to p in $G^{(t)}$ (if a vertex is chosen many times then its multiplicity is counted in Γ). Then, one can easily show (for example, using Chernoff bound) that with high probability at least $1 - 1/n^5$, we have $|\frac{n \cdot \Gamma}{N} - \deg_t(p)| \leq \frac{1}{4} \deg_t(p)$.

Since in general we do not know the value of $\deg_t(p)$, we can estimate it by starting with sample sizes $N_i = c \cdot n \cdot \log n / 2^i$ for integers $i = \lfloor \log n \rfloor$ down to 0, and stop when we obtain for the first time $|\frac{n \cdot \Gamma}{N_i} - 2^i| \leq \frac{1}{4} \cdot 2^i$. One can easily show that (with probability at least $1 - 1/n^4$) the value 2^i is then an approximation of $\deg_t(t)$ to within a factor of 2. Now, once we know an approximation of $\deg_t(p)$ to within a factor of 2 we can easily output the appropriate value of $\widehat{D}(v)$. Finally, it is easy to see that the running time of such an algorithm is $\mathcal{O}(n \log n / \deg_t(p))$ (with probability at least $1 - 1/n^4$). □

6 A sublinear time algorithm for estimating $c^{(t)}$

In this section we describe and analyze our $\widetilde{\mathcal{O}}(n \cdot \zeta^{-3} \cdot \epsilon^{-1} \cdot \rho^{-1})$-time algorithm for estimating the number of connected components $c^{(t)}$ in $G^{(t)}$. The algorithm combines the sampling approach from [6], the graph traversal algorithm Clique-Tree-Traversal described in Section 3, and the sampling algorithm Degree-Estimate used to recognize high degree vertices in the sample.

We present now our algorithm Number-of-Connected-Components (P, t, ζ, ρ, ϵ).

Number-of-Connected-Components (P, t, ζ, ρ, ϵ)

s=0

while running time is less than $T^* = \widetilde{\mathcal{O}}(n \cdot \zeta^{-3} \cdot \epsilon^{-1} \cdot \rho^{-1})$ **do**

  $s = s + 1$; $\beta_s = 0$

  choose a vertex p_s independently and uniformly at random

  choose integer X according to $\mathbf{Pr}[X \geq k] = 1/k$

  $\widehat{D}(p_s)$ = Degree-Estimate(P, t, p_s)

  if $\widehat{D}(p_s) \leq 2X$ **then**

    run Clique-Tree-Traversal (P, p_s, t, ϵ) until one of the following events happens:

      (1) more than X vertices are explored

      (2) more than $\frac{4}{\zeta \cdot \epsilon}$ representative vertices are explored

      (3) the entire connected component in $G^{(t)}$ containing p_s is explored

    if event (3) happened **then** $\beta_s = 1$

output $\hat{c}^{(t)} = \frac{n}{s} \cdot \sum_{i=1}^{s} \beta_i$

We say algorithm Degree-Estimate(P, t, p) *works properly* if it returns a value $\widehat{D}(p)$ with $\frac{1}{2}\widehat{D}(p) \leq \deg_t(p) \leq 2\widehat{D}(p)$ and its running time is $\mathcal{O}(n \cdot \log n / \deg_t(p))$. Notice that by Lemma 5.1, every run of algorithm Degree-Estimate(P, t, p) works properly with probability at least $1 - 1/n^4$. Obviously, we can assume that the overall running time is $o(n^2)$ because otherwise we can simply compute the minimum spanning tree directly. Therefore, with probability at least $1 - 1/n^2$, all runs of algorithm Degree-Estimate(P, t, p) incorporated in algorithm Number-of-Connected-Components $(P, t, \zeta, \rho, \epsilon)$ work properly. Hence, from now on, we shall condition on this fact (that holds with probability at least $1 - 1/n^2$).

Before we proceed with the analysis of the algorithm, we first explain our use of algorithm Degree-Estimate that is needed to decrease the total running time of the algorithm and has no influence on the output value.

If in the ith iteration of algorithm Number-of-Connected-Components procedure Degree-Estimate returns a value $\widehat{D}(p) > 2X$ then we know that $\deg_t(p) > X$. If $\deg_t(p) > X$ then we know that $n_p^t > X$. Hence, our procedure would stop the Clique-Tree-Traversal because of event (1) before event (3) could happen. This would cause $\beta_i = 0$. Therefore, we do not have to invoke the Clique-Tree-Traversal in that case and we can immediately set $\beta_i = 0$.

For the remaining analysis (besides the running time analysis) we can therefore ignore the procedure Degree-Estimate. We can assume that for every sampled vertex p_i, we set $\beta_i = 0$, if algorithm Clique-Tree-Traversal (P, t) stops because of event (1) or (2), or we set $\beta_i = 1$, otherwise.

Our next step is to prove that the expected value of $\hat{c}^{(t)}$ is close to $c^{(t)}$.

Lemma 6.1 (Expectation bound) *If in all calls algorithm Degree-Estimate work properly, then the random variable $\hat{c}^{(t)}$ computed in algorithm Number-of-Connected-Components satisfies the following:*

$$c^{(t)} \;\geq\; \mathbf{E}\left[\hat{c}^{(t)}\right] \;\geq\; c^{(t)} - \frac{\zeta}{2t} \cdot \text{MST} \;\; .$$

Proof : Recall that if we ignore events of type (2), then we have already seen in Section 3 that $\mathbf{E}\left[\hat{c}^{(t)}\right] = c^{(t)}$. Since the introduction of events of type (2) can only decrease the expected value of $\hat{c}^{(t)}$, the inequality $c^{(t)} \geq \hat{c}^{(t)}$ follows.

Now, we prove the second inequality, namely, $\mathbf{E}[\hat{c}^{(t)}] \geq c^{(t)} - \frac{\zeta}{2t} \cdot \text{MST}$. We partition the t-connected components in P into two types. A t-connected component C is of type (II) if there *exists* a vertex $p \in C$ such that the Clique-Tree-Traversal with starting vertex p stops with more than $\frac{4}{\zeta\epsilon}$ representative vertices. Otherwise, a t-connected component is of type (I). The idea behind these two types is that our algorithms always counts connected components of type (I) but it may not count connected components of type (II).

Let K denote the number of connected components of type (II). Then $\text{MST} \geq \frac{K\epsilon t}{2} \cdot \frac{4}{\zeta\epsilon} = \frac{2Kt}{\zeta}$, and hence $K \leq \frac{\zeta}{2t} \cdot \text{MST}$. We also know that

$$\mathbf{E}[\beta_i] \;\geq \sum_{\text{type (I) connected component } C} \mathbf{Pr}[p_i \in C] \cdot \mathbf{Pr}[X \geq |C|] \;=\; \frac{c^{(t)} - K}{n} \;\; .$$

Hence,

$$\mathbf{E}[\hat{c}^{(t)}] \;\geq\; c^{(t)} - K \;\geq\; c^{(t)} - \frac{\zeta}{2t} \cdot \text{MST} \;\; .$$

□

Our next step is to prove a bound on the number of iterations of algorithm Number-of-Connected-Components. Here we will make use of the dispersion $\mathcal{L}(G^{(t)})$ and so our bound will also depend on this value.

Lemma 6.2 (Sample size) *Let $0 < \epsilon < 1/2$. If in all calls algorithm Degree-Estimate work properly, then for certain $T^* = \widetilde{\mathcal{O}}(n\,\zeta^{-3}\,\epsilon^{-1}\,\rho^{-1})$ the number of iterations s of algorithm Number-of-Connected-Components is at least $n\,\zeta^{-2}\,\rho^{-1}/\mathcal{L}(G^{(t)})$, with probability at least $1-\rho$.*

Proof : We assume that a counter T is used in the algorithm to count the running time of the algorithm. Let $T^{(i)}$ denote the value of T at the end of the ith iteration of the **while**-loop in algorithm Number-Of-Connected-Components. We give an upper bound on the expected increase $\Delta T^{(i)} = T^{(i+1)} - T^{(i)}$ of variable T in a single iteration of this loop. We will use a sufficiently large absolute constant α to avoid the use of the big-Oh notation.

We start our analysis with a partition of P into $\mathcal{L}(G^{(t)})$ *clusters* C_j according to the full Clique-Tree-Traversal. There is exactly one cluster for each representative vertex. If a vertex p is no representative vertex then it was explored from some representative vertex q. In this case we assign p to the cluster containing q. We observe that each cluster C_j forms a clique in $G^{(t)}$, because the distance between any two points in C_j is at most $2\,\epsilon\,t$ and $\epsilon < 1/2$. For any vertex p, let C_p denote the cluster that contains p. Notice that $\deg_t(p) \geq |C_p| - 1$.

If all calls to algorithm Degree-Estimate work properly then the test $\widehat{D}(p) < 2X$ rejects every vertex p in a cluster of size greater than $2X$. In this case we increase T by at most $\alpha \cdot n \log n/\widehat{D}(p)$, where α is a constant used to upper bound the constants hidden in the big-Oh notation of the running time of Degree-Estimate. We get

$$\begin{aligned}\Delta T^{(i)} &= T^{(i+1)} - T^{(i)} = \alpha \cdot n \cdot \log n/\widehat{D}(p) \leq 2 \cdot \alpha \cdot n \cdot \log n/\deg_t(p) \\ &\leq 2 \cdot \alpha \cdot n \cdot \log n/(|C_p| - 1) \leq 4 \cdot \alpha \cdot n \cdot \log n/|C_p| \ .\end{aligned}$$

For any vertex that is in a cluster of size smaller than or equal to $2X$, we increase T by at most $\alpha \cdot n/(\zeta\,\epsilon)$. Since the number of clusters is $\mathcal{L}(G^{(t)})$, we clearly have at most $2X \cdot \mathcal{L}(G^{(t)})$ vertices in a cluster of size at most $2X$. Now we observe that in the case $X > n$ the behavior of our algorithm is identical to the case $X = n$. Therefore, we define a random variable $X^* = X$ for $X < n$ and $X^* = n$ for $X \geq n$. We get for fixed value of X^*,

$$\begin{aligned}\mathbf{E}\big[\Delta T^{(i)}\big] &\leq \sum_{p\,:\,C_p \leq 2X^*} \mathbf{Pr}[p_i = p] \cdot \frac{\alpha \cdot n}{\zeta\,\epsilon} + \sum_{p\,:\,C_p > 2X^*} \mathbf{Pr}[p_i = p] \cdot \frac{4\,\alpha \cdot n \cdot \log n}{|C_q|} \\ &= \sum_{p\,:\,C_p \leq 2X^*} \frac{1}{n} \cdot \frac{\alpha \cdot n}{\zeta\,\epsilon} + \sum_{p\,:\,C_p > 2X^*} \frac{1}{n} \cdot \frac{4\,\alpha \cdot n \cdot \log n}{|C_p|} \\ &\leq \frac{2 \cdot \alpha \cdot X^* \cdot \mathcal{L}(G^{(t)})}{\zeta\,\epsilon} + 4 \cdot \alpha \sum_{p \in P} \frac{1}{|C_p|} \leq \frac{6 \cdot \alpha \cdot X^* \cdot \mathcal{L}(G^{(t)}) \cdot \log n}{\zeta\,\epsilon} \ .\end{aligned}$$

Since the choice of X^* is independent of the other choices, the inequality $\mathbf{E}[X^*] \leq \log n$ implies that

$$\mathbf{E}\big[\Delta T^{(i)}\big] \leq \frac{6 \cdot \alpha \cdot \log^2 n \cdot \mathcal{L}(G^{(t)})}{\zeta\,\epsilon} \ ,$$

and hence also

$$\mathbf{E}\big[T^{(i)}\big] \;\leq\; \frac{6\cdot i\cdot \alpha\cdot \log^2 n\cdot \mathcal{L}(G^{(t)})}{\zeta\,\epsilon}\;.$$

Thus we can apply Markov inequality to obtain that for any $\rho > 0$

$$\mathbf{Pr}\big[T^{(i)} \;\geq\; \frac{6\cdot i\cdot \alpha\cdot \log^2 n\cdot \mathcal{L}(G^{(t)})}{\rho\,\zeta\,\epsilon}\big] \;\leq\; \rho\;.$$

We conclude that for $T^* = 6\cdot n\cdot \alpha\cdot \log^2 n\cdot \zeta^{-3}\cdot \epsilon^{-1}\cdot \rho^{-1} = \Theta(n\cdot \log^2 n\cdot \zeta^{-3}\cdot \epsilon^{-1}\cdot \rho^{-1})$ the number of iterations of the **while**-loop is at least $n\cdot\zeta^{-2}\cdot\rho^{-1}/\mathcal{L}(G^{(t)})$ with probability at least $1-\rho$. □

Lemma 6.3 (Concentration bound) *If in all calls algorithm Degree-Estimate works properly, then for the random variable $\hat{c}^{(t)}$ computed by algorithm Number-of-Connected-Components the following bound holds:*

$$\mathbf{Pr}\big[|\hat{c}^{(t)} - \mathbf{E}\big[\hat{c}^{(t)}\big]| \leq \zeta\cdot\max\{c^{(t)}, \mathcal{L}(G^{(t)})\}\big] \;\geq\; 1-2\rho\;.$$

Proof : Similarly to [6], we can upper bound the variance of any single β_i as follows:

$$\mathbf{Var}[\beta_i] \;\leq\; \mathbf{E}\big[\beta_i^2\big] \;\leq\; \mathbf{E}\big[\beta_i\big] \;\leq\; \frac{c^{(t)}}{n}\;,$$

where the last inequality follows from our analysis in the proof of Lemma 6.1 and from Section 3. Let s denote the number of iterations of Number-Of-Connected-Components. Then we have

$$\mathbf{Var}[\hat{c}^{(t)}] \;=\; \Big(\frac{n}{s}\Big)^2\cdot\sum_{1\leq i\leq s}\mathbf{Var}[\beta_i] \;\leq\; \Big(\frac{n}{s}\Big)^2\cdot s\cdot\frac{c^{(t)}}{n} \;=\; \frac{n\,c^{(t)}}{s}\;.$$

Next, by Chebyshev inequality we obtain,

$$\mathbf{Pr}\big[|\hat{c}^{(t)} - \mathbf{E}\big[\hat{c}^{(t)}\big]| \geq \zeta\cdot\max\{c^{(t)}, \mathcal{L}(G^{(t)})\}\big] \;\leq\; \frac{n\cdot c^{(t)}}{s\cdot(\zeta\cdot\max\{c^{(t)}, \mathcal{L}(G^{(t)})\})^2} \;\leq\; \frac{n}{s\cdot\zeta^2\cdot\mathcal{L}(G^{(t)})}\;.$$

Conditioned on the event that $s \geq n\cdot\rho^{-1}\cdot\zeta^{-2}/\mathcal{L}(G^{(t)})$ we obtain,

$$\mathbf{Pr}\big[|\hat{c}^{(t)} - \mathbf{E}\big[\hat{c}^{(t)}\big]| \geq \zeta\cdot\max\{c^{(t)}, \mathcal{L}(G^{(t)})\}\big] \;\leq\; \rho\;.$$

Since $s \geq n\cdot\rho^{-1}\cdot\zeta^{-2}/\mathcal{L}(G^{(t)})$ holds with probability $1-\rho$, by Lemma 6.2 we finally obtain

$$\mathbf{Pr}\big[|\hat{c}^{(t)} - \mathbf{E}\big[\hat{c}^{(t)}\big]| \leq \zeta\cdot\max\{c^{(t)}, \mathcal{L}(G^{(t)})\}\big] \;\geq\; 1-2\,\rho\;.$$

□

Now, we can summarize our entire discussion in this section with the following theorem which immediately implies Theorem 1.

Theorem 2 *Given P, t, ϵ, ζ and $\rho \geq 1/n^2$, algorithm Number-Of-Connected-Components computes in $\widetilde{\mathcal{O}}(n\cdot\zeta^{-3}\cdot\epsilon^{-1}\cdot\rho^{-1})$ time a value $\hat{c}^{(t)}$ that with probability at least $1-3\,\rho$ satisfies the following*

$$(1-\zeta)\cdot c^{(t)} - \zeta\cdot\frac{MST}{\epsilon\,t} \;\leq\; \hat{c}^{(t)} \;\leq\; (1+\zeta)\cdot c^{(t)} + \zeta\cdot\frac{MST}{\epsilon\,t}\;.$$

Proof : Let us first assume that all calls of algorithm Degree-Estimate work properly. Lemma 6.1 gives us

$$c^{(t)} - \frac{\zeta}{2 \cdot t} \text{MST} \;\leq\; \mathbf{E}\big[\hat{c}^{(t)}\big] \;\leq\; c^{(t)} \;.$$

We prove the lemma in two cases, depending on whether $c^{(t)} \geq \mathcal{L}(G^{(t)})$ or $c^{(t)} < \mathcal{L}(G^{(t)})$.

Let us first consider the case $c^{(t)} \geq \mathcal{L}(G^{(t)})$. Then, by Lemma 6.3 we have with probability at least $1 - 2\rho$ the inequality $|\hat{c}^{(t)} - \mathbf{E}\big[\hat{c}^{(t)}\big]| \leq \zeta \cdot c^{(t)}$. Hence,

$$\begin{aligned} \hat{c}^{(t)} &\leq \mathbf{E}\big[\hat{c}^{(t)}\big] + \zeta \cdot c^{(t)} \;\leq\; (1+\zeta) \cdot c^{(t)} \;, \\ \hat{c}^{(t)} &\geq \mathbf{E}\big[\hat{c}^{(t)}\big] - \zeta \cdot c^{(t)} \;\geq\; (1-\zeta) \cdot c^{(t)} - \frac{\zeta}{2 \cdot t} \text{MST} \;. \end{aligned}$$

On the other hand, if $c^{(t)} < \mathcal{L}(G^{(t)})$, then we can use Lemma 4.3 to obtain $\mathcal{L}(G^{(t)}) \leq \text{MST}/(4\,\epsilon\,t)$. This together with Lemma 6.3 imply that with probability at least $1 - 2\rho$ we have

$$\begin{aligned} \hat{c}^{(t)} &\leq \mathbf{E}\big[\hat{c}^{(t)}\big] + \zeta \cdot \mathcal{L}(G^{(t)}) \;\leq\; c^{(t)} + \zeta \cdot \frac{\text{MST}}{4 \cdot \epsilon \cdot t} \;, \\ \hat{c}^{(t)} &\geq \mathbf{E}\big[\hat{c}^{(t)}\big] - \zeta \cdot \mathcal{L}(G^{(t)}) \;\geq\; c^{(t)} - \zeta \cdot \frac{\text{MST}}{2 \cdot t} - \zeta \cdot \frac{\text{MST}}{4 \cdot \epsilon \cdot t} \;\geq\; c^{(t)} - \frac{\zeta}{\epsilon \cdot t} \text{MST} \;. \end{aligned}$$

Finally, since all calls of algorithm Degree-Estimate work properly with probability at least $1 - 1/n^2$, the lemma follows. □

7 Reducing general case to all distances being powers of $(1+\epsilon)$

In all previous analyses we assumed that all distances are powers of $(1+\epsilon)$. In this section we justify this assumption and show how our analysis can be extended to the general case when all distances are arbitrary real numbers in the interval $[1, 2n/\epsilon]$.

If we have arbitrary distances then our analysis has to be modified because the identity (1) does not hold anymore and, more importantly, Lemma 4.1 is invalid. The first problem can can be easily fixed by observing that the use of identity (1) to arbitrary distances introduces at most a $(1+\epsilon)$-factor error term. The other problem is slightly more complex because the set of vertices E explored by the Clique-Tree-Traversal (1) depends on the starting vertex v and (2) might be different from the connected component $C_v^{(t)}$ containing v in $G^{(t)}$.

Before we show how to deal with this problem let us consider the following modification of the Clique-Tree-Traversal. In line 2 of the algorithm we consider edges of length $(1+\epsilon) \cdot t$ instead of edges of length t.

Clique-Tree-Traversal* (P, p, t, ϵ)
$R = \{p\}$; $E = \{p\}$; $U = P \setminus \{p\}$
while there is an edge $e = (p, q) \in E^{((1+\epsilon)\cdot t)}$ with $p \in R$ and $q \in U$ **do**
  let (p, q) be the shortest such edge
  $E = E \cup \{q\}$; $U = U \setminus \{q\}$
  if $d(p, q) \geq \epsilon\, t$ **then** $R = R \cup \{q\}$

Now we can easily prove that independent of the starting vertex v the inequality $C_v^{(t)} \subseteq E \subseteq C_v^{((1+\epsilon)\cdot t)}$ holds. Using this observation we obtain with small modifications in the proof the following result to be held with probability at least $1-\rho$:

$$(1-\zeta)\cdot c^{((1+\epsilon)^i)} - \zeta\cdot\frac{\text{MST}}{\epsilon\cdot(1+\epsilon)^i} \;\le\; \hat{c}^{((1+\epsilon)^i)} \;\le\; (1+\zeta)\cdot c^{((1+\epsilon)^{i+1})} + \zeta\cdot\frac{\text{MST}}{\epsilon\cdot(1+\epsilon)^i}\;. \qquad (3)$$

With this inequality we can prove that our estimator $n - W + \epsilon\cdot\sum_{i=0}^{\log_{1+\epsilon} W-1}(1+\epsilon)^i\cdot\hat{c}^{((1+\epsilon)^i)}$ is a $(1+3\,\epsilon)$-approximation of MST. Adjusting the constants in the proof gives a $(1+\epsilon)$-approximation with $\widetilde{\mathcal{O}}(n/\epsilon^8)$ running time.

8 Every approximation algorithm of MST within *any* factor requires time $\Omega(n)$

It is easy to see that no algorithm with $o(n)$ running time can approximate the cost of the minimum spanning tree within *any* factor. For a given approximation factor B let us consider two graphs G_1 and G_2. G_1 consists of a clique of $n-1$ vertices having mutual distance 1 and a single outlier with distance $2\,B\,n$ to each other vertex. In graph G_2 the distance between every pair of vertices is 1. Clearly, the minimum spanning tree of graph G_2 has cost $n-1$ while the minimum spanning tree of graph G_1 has cost $n-2+2\,B\,n$. In order to distinguish between the two graph one has to find the single outlier, what cannot be achieved in time $o(n)$ with constant confidence probability. This yields the following easy claim.

Theorem 3 *No $o(n)$-time algorithm can approximate the weight of the minimum spanning tree within any factor.* □

References

[1] N. Alon, S. Dar, M. Parnas, and D. Ron. Testing of clustering. *Proc. 41st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 240–250, 2000.

[2] T. Batu, F. Ergün, J. Kilian, A. Magen, S. Raskhodnikova, R. Rubinfeld, and R. Sami. A sublinear algorithm for weakly approximating edit distance. *Proc. 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 316–324, 2003.

[3] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1): 67–90, January 1995.

[4] B. Chazelle. A minimum spanning tree algorithm with inverse-Ackermann type complexity. *Journal of the ACM*, 47(6): 1012–1027, November 2000.

[5] B. Chazelle, D. Liu, and. Magen. Sublinear geometric algorithms. *Proc. 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 531–540, 2003.

[6] B. Chazelle, R. Rubinfeld, and L. Trevisan. Approximating the minimum spanning tree weight in sublinear time. *Proc. 28th Annual International Colloquium on Automata, Languages and Programming (ICALP)*, pages 190–200, 2001.

[7] A. Czumaj, F. Ergün, L. Fortnow, A. Magen, I. Newman, R. Rubinfeld, and C. Sohler. Sublinear-time approximation of Euclidean minimum spanning tree. *Proc. 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 813–822, 2003.

[8] D. Eppstein. Spanning trees and spanners. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, chapter 9, pages 425–461. Elsevier Science B.V., 1997.

[9] A. Frieze and R. Kannan. The regularity lemma and approximation schemes for dense problems. *Proc. 37th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 12–20, 1996.

[10] A. Frieze, R. Kannan, and S. Vempala. Fast Monte-Carlo algorithms for finding low-rank approximations. *Proc. 39th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 370–378, 1998.

[11] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4): 653–750, July 1998.

[12] P. Indyk. A sublinear time approximation scheme for clustering in metric spaces. *Proc. 39th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 154–159, 1998.

[13] P. Indyk. Sublinear time algorithms for metric space problems. *Proc. 31st Annual ACM Symposium on Theory of Computing (STOC)*, pages 428–434, 1999.

[14] D. R. Karger, P. N. Klein, and R. E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *Journal of the ACM*, 42(2): 321–328, March 1995.

[15] N. Mishra, D. Oblinger, and L. Pitt. Sublinear time approximate clustering. *Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 439–447, 2001.

[16] S. Pettie and V. Ramachandran. An optimal minimum spanning tree algorithm. *Journal of the ACM*, 49(1): 16–34, 2002.

[17] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, Berlin, 2001.

[18] A. C. Yao. On Constructing minimum spanning trees in k-demensional spaces and related problems. *SIAM Journal on Computing*, 11: 721–736, 1982.