A linear time approximation scheme for Euclidean TSP

Yair Bartal
Hebrew University
Email: yair@cs.huji.ac.il

Lee-Ad Gottlieb
Ariel University
Email: leead@ariel.ac.il

***Abstract*—The Traveling Salesman Problem (TSP) is among the most famous NP-hard optimization problems. The special case of TSP in bounded-dimensional Euclidean spaces has been a particular focus of research: The celebrated results of Arora [Aro98] and Mitchell [Mit99] – along with subsequent improvements of Rao and Smith [RS98] – demonstrated a polynomial time approximation scheme for this problem, ultimately achieving a runtime of $O_{d,\varepsilon}(n \log n)$.**

In this paper, we present a linear time approximation scheme for Euclidean TSP, with runtime $O_{d,\varepsilon}(n)$. This improvement resolves a 15 year old conjecture of Rao and Smith, and matches for Euclidean spaces the bound known for a broad class of planar graphs [Kle08].

***Keywords*-Computations on discrete structures; Geometrical problems and computations;**

I. Introduction

Among all NP-complete problems, the *Traveling Salesman Problem (TSP)* stands out as fundamental and is studied extensively. Indeed, numerous articles and even whole books ([Rei94], [LLKS85], [GP02], [ABCC07]) are devoted to TSP, studying various algorithms for different families of instances. In fact, some of the most basic techniques in combinatorial optimization were devised to tackle TSP, including for instance cutting planes. The input for (the optimization version of) TSP is a complete graph, whose vertex set we denote by S, together with edge-weights $w(\cdot, \cdot)$ that are nonnegative and symmetric,[1] and the goal is to find a closed tour of S of minimum (total) weight, where a tour is simply a permutation of S, i.e. it visits every vertex exactly once.

A prominent special case of TSP – called *Euclidean TSP* – is where the points are given as d-dimensional vectors equipped with the ℓ_2 norm. Celebrated results of Arora [Aro98] (for d-dimensional space) and Mitchell [Mit99] (for the Euclidean plane) prove that Euclidean TSP admits a Polynomial-Time Approximation Scheme (PTAS). A PTAS is an algorithm that provides a $(1+\varepsilon)$–approximation for every *fixed* $\varepsilon > 0$. For every constant $\varepsilon > 0$, the runtime is polynomial in n. The basic techniques employed by Arora include the imposition of a random quadtree on the point set, and a proof that there exists a low-cost tour for S which crosses each quadtree cell a small number of times, and then only at a small number of previously designated quadtree portals. Given such a configuration, a brute-force dynamic program is employed to construct the tour on the quadtree in a bottom-up fashion. The final algorithm achieved a runtime of $n \log^{O(\sqrt{d}/\varepsilon)^{d-1}} n$. Subsequently, Rao and Smith noted that the solution tour may be restricted to the edges of a low-weight spanner graph of the input set, and this allows one to eliminate the requirement that the tour cross the random quadtree only at previously designated portals. They thereby achieved runtime $2^{(d/\varepsilon)^{O(d)}} + (d/\varepsilon)^{O(d)} n \log n$. Complementing these results, Trevisan [Tre00] showed that TSP in Euclidean metrics of dimension $\log n$ is NP-hard to approximate to within some constant $c > 1$. It is therefore not surprising that the running time of the aforementioned PTAS is doubly-exponential in the dimension.

Rao and Smith [RS98] further posited that that a stronger result may be possible (see also [Cha08], [BE12], [Epp12]). We may encapsulate their conjecture in the following question:

Question 1. *Does Euclidean TSP admit an approximation algorithm with* linear *dependence on the set size?*

It turns out that in some common models of computation, the answer to Question 1 is in fact negative. For example, results of Das *et al.* [DKS97] and Young [You13] demonstrate a $\Omega(n \log n)$ time lowerbound for 1-dimensional Euclidean TSP in the decision tree model. Indeed, the conjecture of Rao and Smith was expressed specifically in the real RAM model, augmented with the floor or mod functions as atomic operators (i.e., basic $O(1)$ time operations).

A. Results

Our central contribution is a linear time approximation scheme for Euclidean TSP.

Theorem I.1. *Given a set S of d-dimensional points with integral coordinates realizing the range $[0, \lceil \frac{n\sqrt{d}}{\varepsilon} \rceil]$, there exists a randomized algorithm that with probability $1 - e^{-O_d(n^{1/3d})}$ computes a $(1+\varepsilon)$-approximation to the optimal tour for S in time $2^{(d/\varepsilon)^{O(d)}} n$ in the integer RAM model.*

The statement of Theorem I.1 assumes that the input vectors are bounded integral values: $S \subset \{0, \ldots, R\}^d$ for $R = \lceil \frac{n\sqrt{d}}{\varepsilon} \rceil$ (though we could actually tolerate any

Work supported in part by Israel Science Foundation grant #1609/11

[1] Formally, $w(x, y) = w(y, x) \geq 0$ for all $x, y \in S$.

$R = 2^{(d/\varepsilon)^{O(d)}} n$). That is, we assume the data is not padded with extraneous information unnecessary for achieving a $(1+\varepsilon)$-approximation (similar to Arora's rounding the points to a grid [Aro98]). If the data is not given in this integral form, we may round the data into this form using the floor or mod functions – assuming these functions are atomic operations, the rounding can be done in $O(dn)$ total time. We note that this *one-time* rounding step is the only non-trivial operation utilized in this paper, and we further do not make use of bit-wise operators. Hence, we resolve the conjecture of Rao and Smith [RS98] in the affirmative:

Corollary I.2. *Given a set S of d-dimensional points, there exists a randomized algorithm that with probability $1 - e^{-O_d(n^{1/3d})}$ computes a $(1+\varepsilon)$-approximation to the optimal tour for S in time $2^{(d/\varepsilon)^{O(d)}} n$ in the real-RAM model with atomic floor or mod operators.*

B. Techniques

We build upon the framework of [Aro98], [RS98], and will assume that the reader has basic familiarity with these works. Our algorithm includes the following linear time techniques: (i) A dynamic graph which approximates the true weight of the MST for the input set, over all localities (Section IV). (ii) A procedure that decompose the input set into a group of sets, each of which has a sparse minimum spanning tree (Section V). This construction was inspired by the recent decomposition scheme of [BGK12], although that scheme required superlinear time. (iii) A clustering for the sparse sets, in Section VI. This allows simultaneous calculation of a low weight spanner and low distortion tour.

We first show that a shallow hierarchy for point sets can be constructed very quickly. Using this hierarchy, we break up the input set into smaller sets with very favorable properties – that the MST of these sets are everywhere sparse. Hence, these sets have the property that their low-weight spanners are also everywhere sparse. Finally, we show how to build a clustering which is crossed few times by the low weight spanners. Utilizing the standard dynamic programming algorithm, we can immediately derive a low weight tour which is restricted to the edges of the spanner.

Related work: Arora's geometric approach was subsequently employed for other Euclidean problems in [CL98], [ARR98], [CLZ02], [KR07]. A linear time approximation scheme for TSP on a class of planar graphs was given in [Kle08].

A series of papers culminated in a PTAS for metric spaces with bounded doubling dimension [Tal04], [BGK12]. Chan and Gupta [CG08] gave an algorithm for TSP that runs in sub-exponential time in a larger family of instances, in which an alternative notion of dimension is assumed to be bounded. Further extension of the algorithms of [Aro98], [Tal04] to the problem of TSP with neighborhoods (under mild conditions) include [Mit07] and [CE11].

II. Preliminaries

Here we present background information.

Euclidean properties: For a d-dimensional point set S, we will use the notation $B(u,s) \cap S$ to refer to the points of S contained in the ball centered at u with radius s; that is, all points of S within distance s of u. When S is understood from context, we may refer just to $B(u,s)$. $B^*(u,s)$ is the edge set of the full graph on $B(u,s)$.

It is well known that if S contains points with minimum interpoint distance 1, then $|B(u,s) \cap S| = s^{O(d)}$ (when $s \geq 2$). We will refer to this fact as the Packing Property of Euclidean space.

TSP tours: Throughout, a *tour* W is a finite sequence of points; by convention, it is undirected, and may visit a point more than once. A *transition* in W is a pair of successive points in the sequence, which may be viewed as an edge in the complete graph on S. Let $w(x,y)$ for $x, y \in S$ be the length (or *weight*) of an edge connecting x and y. A *closed* tour is defined in the natural way by adding a transition between the last and first points in the sequence.

The *weight* (or *length*) of a multiset M of transitions is defined as $w(M) \stackrel{\text{def}}{=} \sum_{(x,y)\in M} w(x,y)$. This notation naturally extends to a tour W, by viewing W as sequence of transitions, hence $w(W)$ represents the total length of the tour W. Let $\mathrm{OPT}(S)$ denote the minimum weight TSP tour on the set S. It is well-known that $w(\mathrm{MST}(S)) \leq w(\mathrm{OPT}(S)) < 2w(\mathrm{MST}(S))$. For Euclidean set S, we also have $w(\mathrm{MST}(S)) = O(|S|^{1-1/d} \operatorname{diam}(S))$ [Aro98].

Point hierarchies: Similar to what was described in [GGN06], [KL04], a subset of points $X \subseteq Y$ is an s-net of Y if it satisfies the following properties:

(i) Packing: For every $x, y \in X$, $d(x,y) \geq s$.
(ii) Covering: Every point $y \in Y$ is strictly within distance s of some point $x \in X$: $d(x,y) < s$.

The previous conditions require that the points of X be spaced out, yet nevertheless cover all points of Y. A point in Y covering a point in X is called a *parent* of the covered point; this definition allows for a point to have multiple parents.

A hierarchy $\mathcal{H}$ for set S is composed of discrete center sets, where each level of the hierarchy is a discrete center set of the level beneath it. For each $i = 0, \ldots, P$ (where $P := \lceil \log \operatorname{diam}(S) \rceil$), fix $H_i \subseteq S$ to be an 2^i-net of S, called the net of *level* i, or of *scale* 2^i. Notice that the bottom hierarchical level H_0 contains all points, and the top level H_P contains only a single point. The hierarchy may be augmented with neighbor links: Each point $x \in H_i$ records what points of H_i are within distance $b \cdot 2^i$ of x – these are the b-neighbors of x. By the packing property of Euclidean spaces, a point may have $b^{O(d)}$ b-neighbors. We will need the following lemma:

Lemma II.1. *If two points $u, v \in H_j$ are b-neighbors (for $b \geq 8$), then all distinct respective ancestors $u', v' \in H_i$*

($i > j$) are also b-neighbors.

Proof: We have $d(u,u') \leq \sum_{k=j}^{i} 2^k < 2 \cdot 2^i$, and similarly $d(v,v') < 2 \cdot 2^i$. It follows that $d(u',v') \leq d(u',u)+d(u,v)+d(v,v') < 4 \cdot 2^i + b \cdot 2^j \leq (4+\frac{b}{2})2^i \leq b2^i$. ∎

To save space in the hierarchy, we use the standard compression scheme found in [CG06]: First assign each point in H_i as a child of only one parent in H_{i+1}. (If a copy of the smae point exists in H_{i+1}, then that copy will be the assigned parent.) Now, beginning at $i = 1$ and proceeding upwards, any point of H_i that has no b-neighbors and only one child in H_{i-1} is represented implicitly. This scheme results in a *compressed* hierarchy $\tilde{\mathcal{H}}$, whose points and neighbor links can be stored in $b^{O(d)}n$ space. For a point $x \in H_i$ which *survives* (is stored explicitly) in the compressed hierarchy $\tilde{\mathcal{H}}$, we may refer to its lowest surviving ancestor as its parent. Note that lemma II.1 implies that if $u, v \in H_i$ are b-neighbors, then these points and all their distinct respective ancestors must survive as well.

From a compressed hierarchy, one can extract a net-tree T by representing each surviving point occurrence in each level H_i as a node in tree level T_i, and placing an edge between nodes in T which represent parent-child pairs in $\tilde{\mathcal{H}}$. Note that if a single point $v \in S$ appears multiple times in the compressed hierarchy, then there will be multiple nodes in T corresponding to v.

Both the compressed hierarchy and net-tree T support deletions of an entire subtree T' in $2^{O(d)}|T'|$ time, by simply removing these points from the tree and maintaining in T only the root of T'. (This procedure assures that no Steiner points remain in the hierarchy.) The total size of $\tilde{\mathcal{H}}$ and T remains $b^{O(d)}n$, where S is the updated point set.

Quadtrees: Let $R \in \mathbb{N}$ be a power of 2, set $L = \{0, 1, 2, \ldots, R\}$, and let $S = L^d$ be a set of d-dimensional vectors. Let $t = \log_2 R$. Define the quadtree Q of S rooted at the origin as follows: The top level of Q – level $i = t$ – contains a single cell (hypercube) of side-length R rooted at the origin, and covers all points. The next level Q_{i-1} contains 2^d child cells of side-length $\frac{R}{2}$, created by partitioning the larger cell using d orthogonal $(d-1)$-dimensional hyperplanes. Each level Q_j is created by partitioning the cells of level Q_{j+1} into child cells in an analogous manner. The bottom level 0 contains cells of side-length 1, each covering at most one point.

A *compressed quadtree* can be derived from a quadtree as follows: First, all empty cells (those containing no points of S) are removed. Then, beginning from the second lowest quadtree level and moving up, any cell with only a single child is removed, and the child is linked as a child of the removed cell's parent. The compressed quadtree has size $O(n)$.

Graph spanners: A graph R is a $(1+\delta)$-stretch spanner of graph G if R is a subgraph of G that contains all nodes of G (but not all edges), and $d_R(u,v) \leq (1+\delta)d_G(u,v)$ for all $u, v \in G$, where $d_G(u,v)$ ($d_R(u,v)$) denotes the shortest path distance between u and v in G (R). As in [KL04], [GGN06], a $(1+\delta)$-stretch spanner for the full graph G on S can be derived from the hierarchy of S by adding an edge between any point pair $x, y \in H_i$ with $d_G(x,y) \leq \frac{2}{\delta}2^i$. We call this edge an H_i level edge. Then the resulting spanner contains only $\delta^{-O(d)}n$ edges. We call this spanner the *complete hierarchical spanner*. (Other spanner constructions may be found in [Vai91], [Sal91], [CK95], [Soa94], [AMS99], [BGM04].)

The weight of the complete hierarchical spanner may be quite high. To produce a light weight spanner, we may prune the spanner edges via the greedy algorithm of [DN97]: The greedy algorithm takes a $(1+\delta)$-stretch Euclidean spanner R, and orders the edges of R by increasing length. It then builds a $(1+\delta)^2$-stretch spanner R' by taking each edge (u,v) in turn, and adding it to P' only if $d_{R'}(u,v) \geq (1+\delta)d_R(u,v)$ (where $d_{R'}(u,v)$ denotes the distances on R' before the addition of the new edge). It follows from the work of [DN97] that the resulting spanner has weight $\delta^{-O(d)}w(\mathrm{MST}(S))$ (see also [GLN02]). We call this spanner the *greedy hierarchical spanner*.

Net-Respecting Tours: A tour W is said to be *net respecting (NR)* relative to a given hierarchy $\{H_i\}_{i=0}^{\lceil \log R \rceil}$ and value $\varepsilon > 0$, if for every transition in W, say of length ℓ, both of its endpoints belong to H_i for i such that $s^i \leq \varepsilon\ell < s^{i+1}$. We will find it convenient to view the edge as connecting the *occurrences* of the endpoints in the single level H_i. (When $\ell < \frac{1}{\varepsilon}$, it suffices to connect H_0 level points; in this case the hierarchy implicitly contains levels H_i for all $i < 0$, and like H_0 these nets contain all points of S.) We denote by $\mathrm{OPT}^{NR}(S)$ a minimum length net-respecting tour on S (for some ε). Any tour restricted to the edges of the $(1+\varepsilon)$-greedy hierarchical spanner is net-respecting.

III. Fast hierarchy construction

In this section, we show that given a set $S \subseteq L^d$, where $L = \{0, 1, 2, \ldots R\}$, we can construct for S a hierarchy with b-neighbor links (for constant b) in time $O_{d,\varepsilon}(R+n)$. The first step is to use a shuffle to construct a quadtree for S in linear time, as in the work of Chan [Cha08] (see also [Gar82], [Har10]). This construction assumes simple bit-wise operators, which we can precompute: By bucketing points in an array of integral range $[0, R]$, we can store each integer in groups of consecutive words of size $\frac{\log n}{3}$, and for each word compute its bit-wise operator with all other words, in time $O(\log n)$ per word pair. The results are stored in a look-up table, and this can all be done in time and space $O_{d,\varepsilon}(R+n)$. It remains only to prove the following:

Lemma III.1. *Given a compressed quadtree Q for S, a hierarchy may be extracted in $d^{O(d)}|S|$ time.*

Proof: We first expand the quadtree Q, so that for every uncompressed cell in Q, all its neighboring cells are uncompressed as well. This can be done in $2^{O(d)}|S|$ time and space. We then build a hierarchy for S in a bottom-up fashion: All points appear in H_0, and H_1 is constructed by including any point not within distance 2 of an already included point. If we associate each point in H_1 with its corresponding containing cell in Q_1, we can determine if a nearby point has already been included, all in $d^{O(d)}$ time (since the cell diagonal has length $\sqrt{d}$ times the cell side-length). Similarly, we include a point of H_i in level H_{i+1} if no H_i point within distance 2^{i+1} has already been included in H_{i+1}. Since a hierarchy possesses $O(n)$ points, the total construction can be done in time $d^{O(d)}|S|$. ∎

IV. Dynamic approximate MST

In this section we show that given point set S and its compressed hierarchy (Section III), one can construct in linear time a graph G which spans S, and whose weight approximates the weight of MST over all local neighborhoods (Lemma IV.1). Further, like the hierarchy and net-tree, graph G can be efficiently maintained under subtree deletions.

Graph construction: The construction of G begins at the bottom level of the hierarchy, and then proceeds iteratively to the next level. We maintain the invariant that at each level H_i, all H_i hierarchical points within distance $c \cdot 2^i$ are in the same component, for $c := 6$. At the end of the algorithmic run on H_i, either a close pair in H_i are connected directly by an H_i level edge, or we have determined that there exists some path between them on lower level edges. In the former case, we say that the points are *explicitly* connected in level H_i; otherwise they are *implicitly* connected.

The algorithm first considers points in H_0, and adds an edge between any pair within distance c. We call the resulting graph G_0. We then consider each level in turn, and all point pairs $u, v \in H_i$ within distance $c \cdot 2^i$. The algorithm inspects all H_{i-1}-net points within distance $2c \cdot 2^i$ of one endpoint, say u, to see if there exists a path among these net points connecting u and v: This path may be formed from both explicit and implicit H_{i-1} connections. If a path is found, we record that u and v are implicitly connected. Otherwise, we add an H_i level edge between u and v. Let G_i be the current graph, consisting of all level H_j edges for $j \leq i$. The final graph is $G = G_{\lceil \log R \rceil}$. We prove the following lemma:

Lemma IV.1. *Graph G possesses the following properties:*

(i) *G can be constructed in time $2^{O(d)}n$, supports the deletion of the points of a hierarchy subtree T from S in $2^{O(d)}|T|$ time, and occupies space $2^{O(d)}|S|$.*
(ii) $w(G \cap B^*(u, 5c \cdot 2^i)) \geq w(\mathrm{MST}(B(u, c \cdot 2^i)))$.
(iii) $w(G \cap B^*(u, c \cdot 2^i)) \leq 2^{O(d)} w(\mathrm{MST}(B(u, 2c \cdot 2^i))$.

Proof: We prove each item in turn.

(i): The construction time and space is immediate from the size of the hierarchy and the packing property. When the points of a hierarchy subtree are removed from S, we maintain only the subtree root in S. Beginning at the bottom hierarchical level, for each point removed from H_i we determine if any close point pair from H_{i+1} (remaining in S) were connected implicitly only via a path through the deleted point. If so, an edge must be added between the pair, and the are connected explicitly. This all can be done in time $2^{O(d)}|T|$.

(ii): By construction of G, any H_i point pair within distance $c2^i$ is connected by a direct edge, or by some edge path in G_{i-1} within distance $\sum_{k=0}^{j} 2c \cdot 2^i < 4c \cdot 2^i$ of either point of the pair. It follows that all points of $B(u, c \cdot 2^i)$ are covered by some connected subgraph of $G \cap B^*(u, 5c \cdot 2^i)$. Then the lemma follows from the optimality of $\mathrm{MST}(B(u, c \cdot 2^i))$, and the fact that $B(u, c \cdot 2^i)$ is a subset of $B(u, 5c \cdot 2^i)$.

(iii): We claim that if $u, v \in H_i$ are connected by a direct edge in G_i, then the $(1+\frac{2}{c})$-stretch greedy hierarchical spanner for S must possess a H_i-level edge with endpoints within distance $2c \cdot 2^i$ of u. To see this, consider the complete hierarchical $(1 + \frac{2}{c})$-stretch spanner R restricted to the hierarchy of $B(u, 2 \cdot 2^i)$. By construction, R connects all H_j point pairs which are within distance $c \cdot 2^j$.

Now, the fact that u and v must be connected in G_i implies that there does not exist any path connecting them in G_{i-1} on lower level edges within distance $2c \cdot 2^i$ of u or v. It follows that there does not exist such a path on the full hierarchical spanner, and so there cannot exist such a path on R. Then R must contain an H_i level edge within distance $2c \cdot 2^i$ of u or v, or else the shortest path between u and v in R has length at least $3c \cdot 2^i$, which is greater than $(1+\frac{c}{2})$ times the true distance from u to v. The lemma then follows from the weight guarantee of R, coupled with the fact that an H_i level edge can be within close proximity of only $2^{O(d)}$ other H_i level points. ∎

V. Removing dense areas

Having constructed the hierarchy $\mathcal{H}$ and spanning graph G, we show in this section how to split S into sparse subsets – subsets whose MST is not much heavier than the diameter of the subset. We will prove the following decomposition theorem:

Theorem V.1. *Given set S and its associated compressed hierarchy $\mathcal{H}$, in time $O(|S|)$ we can split S into sets $S_0, \ldots, S_m \subset S$, $\cup_i S_i = S$, with the following properties, for some $q = (d/\varepsilon)^{\Theta(d)}$:*

(i) *Sparsity: For each S_i, and every $u \in S_i$, $u \in H_j$ we have $w(\mathrm{MST}(B(u, 2^j) \cap S_i)) \leq 2^{O(d)} q 2^j$.*
(ii) *Tour cost: The subset tours overlap to form a single connected tour for all of S, and $\sum_i \mathrm{OPT}^{NR}(S_i) \leq (1+\varepsilon)\,\mathrm{OPT}^{NR}(S)$.*

(iii) *Size:* $\sum_i |S_i| \leq (d/\varepsilon)^{O(d)}|S|$.

The rest of this section is devoted to proving Theorem V.1. We will first present in Section V-A a preliminary lemma concerning the relationship between a local MST and global one, and then present the actual proof in Section V-B.

A. Preliminary lemma

In this section, we will show that if a ball is partitioned into a series of rings (or annuli), then the sum of the weights of the MST for each ring is not much greater than the weight of the MST of the ball.

Given a hierarchy, define the annulus $A(u, s_1, s_2)$ to include points of $B(u, s_2)$ not in $B(u, s_1)$. Define a δ-ball $B^\delta(u, 2^i)$ for $\delta < 1$ to include all $H_{i-\lceil \log(1/\delta)\rceil}$ level points within distance 2^i of u, as well as the entire subtree rooted at each of these points. Define the δ-annulus $A^\delta(u, s_1, s_2)$ to include the points of $B^\delta(u, s_2)$ not in $B^\delta(u, s_1)$. We have the following lemma:

Lemma V.2. *For any ball $B^\delta(u, s)$, fix annuli diameter δ, and define the k-th annuli to be $A^\delta(u, k\delta s, (k+1)\delta s)$. Then $\sum_{k=0}^{\lceil 1/\delta\rceil - 1} w(\mathrm{MST}(A^\delta(u, k\delta s, (k+1)\delta s)) \leq 12 w(\mathrm{MST}(B^\delta(u, s))) + \delta^{-O(d)} s$.*

Proof: The first step is to connect all $H_{i-\lceil \log(1/\delta)\rceil}$ level points in via a spanning graph. For each annulus, we choose an arbitrary point in the annulus and connect it to all points in the annulus. We then connect the $\lceil \frac{1}{\delta} \rceil$ different annuli together via these points. The resulting spanning graph has weight $\delta^{-O(d)} s$.

Now consider some fixed annulus $A = A^\delta(u, p\delta s, (p+1)\delta s)$, and we will show that the cost of constructing a spanning graph connecting the annulus points to each other or to other points of $H_{i-\lceil \log(1/\delta)\rceil}$ can be charged to the edges of $\mathrm{MST}(S)$ inside the larger annulus $\tilde{A} = A^\delta(u, (p-6)\delta s, (p+7)\delta s)$. Then the lemma follows by summing over all annuli.

Take the intersection of $\mathrm{MST}(S)$ with the edges of the full graph on A, and call this intersection edge set E. Let $E_1, \ldots, E_h \subset E$ be maximal connected edge paths in E; we will connect these edge paths into a spanning graph for A. Now, if edge sets E_t, E_{t+1} are connecting by a path of $\mathrm{MST}(S)$ contained completely within $\tilde{A}$, then we connect the last point of E_t to the first point of E_{t+1}, and charge the edge to the path of $\mathrm{MST}(S)$ in $\tilde{A}$. If edge sets E_t, E_{t+1} are connecting by a path exiting $\tilde{A}$, then we connect the last point in E_t and the first point in E_{t+1} to their closest $H_{i-\lceil \log(1/\delta)\rceil}$ level points, and charge the two new edges of total length at most $4\delta s$ to the part of the exiting path found at distance range $[p - 4\delta s, (p+5)\delta s)]$ from u. Since every point of S is within distance $2\delta s$ of an ancestor in $H_{i-\lceil \log(1/\delta)\rceil}$, the charged path has total length at least $4\delta s$. The lemma follows. ∎

B. Partitioning algorithm

To complete Theorem V.1, we will need the following lemma. First, a few definitions: The *size* of a point occurrence $v \in H_i$ is the number of nodes in the subtree of T rooted at the node corresponding to $v \in H_i$. The weight of a point occurrence $v \in H_i$ with respect to G is the sum of the weights of H_i edges in G incident on v. The weight with respect to G of the subtree rooted at $v \in H_i$ is the sum of weights of all points in the subtree.

Lemma V.3. *Given a point set S and its compressed hierarchy $\tilde{\mathcal{H}}$, let H_i be the lowest hierarchical level containing heavy subtrees of weight $q2^i$ or greater. Assume we are given $G_i \subset G$, along with a point $v \in H_i$ which is the root of a heavy subtree. Then S can be split into two* intersecting *sets S_1 and S_2 with the following properties, for any fixed $\varepsilon < \frac{1}{8}$:*

(i) *Sparsity:* $w(\mathrm{MST}(S_2)) \leq 2^{O(d)} q \operatorname{diam}(S_2)$.
(ii) *Tour cost:* $w(\mathrm{OPT}^{NR}(S_1)) + w(\mathrm{OPT}^{NR}(S_2)) \leq w(\mathrm{OPT}^{NR}(S)) + \varepsilon w(\mathrm{OPT}^{NR}(S_2))$.
(iii) *Split size:* $S_1, S_2 \subsetneq S$. $|S_1 \cap S_2| = (d/\varepsilon)^{O(d)}$, *and S_1 and S_2 are covered by respective hierarchies $\tilde{\mathcal{H}}_1, \tilde{\mathcal{H}}_2 \subset \tilde{\mathcal{H}}$, $|\tilde{\mathcal{H}}_1 \cap \tilde{\mathcal{H}}_2| \leq (d/\varepsilon)^{O(d)}$.*
(iv) *Runtime: S_2 can be located in time $O(|S_2|)$.*

Proof: Recall that all descendants of $v \in H_i$ are within distance $2 \cdot 2^i$ of v. We will take S_2 to be all points of some ball $B^\delta(v, s)$, where $\delta = \frac{\varepsilon}{d}$. To find an appropriate value of s, we define $s_k := 2(1+k/\delta)\cdot 2^i$, and for each $k = 0, \ldots, \delta - 1$ in turn, we compute the weight of the MST of the annulus $A^\delta(v, s_k, s_{k+1})$ and of the ball $B^\delta(v, s_k)$. The computation uses the MST algorithm of Fredman and Willard [FW94], which in our setting runs in deterministic linear time (see also [DRK95], [CRT05]). This procedure terminates once we encounter a relatively light weight annulus – one with MST weight at most ε times the weight of the MST of the inner ball. Below, we will prove that a relatively light-weight annulus must exist.

Upon terminating at a value k, we set S_2 to be the (larger) ball $B^\delta(v, s_{k+1})$. The points of S_2 are split off of S, and the remaining set is $S_1 = S - S_2$. We also add to S_1 copies of all the $H_{i-\lceil \log(1/\delta)\rceil}$ level points in S_2. The number of duplicated points is $(d/\varepsilon)^{O(d)}$. Also, since S_2 contains the entire subtree of v, $w(\mathrm{MST}(S_2)) > q2^i$. We prove each item in turn:

(i): H_i is the lowest level containing heavy subtrees, and by the packing property each point of H_i can be the parent of at most $2^{O(d)}$ points of H_{i-1}. It follows that the weight of the MST of the ball $B^\delta(v, s_0)$ is bound by $2^{O(d)} q 2^i$. Likewise, the weight of the MST of any ball $B^\delta(v, s_k)$ is bound by $2^{O(d)} q 2^i$ as well.

(ii): To prove this, we will take the optimal net-respecting tour for S, split it into separate tours for S_1 and S_2, and then patch these respective tours. Consider first the

long edges of length at least $\frac{\varepsilon}{\delta}2^i$ cut by the split. Since the tour for S was net-respecting, these edges are incident on $H_{i-\lceil\log(1/\delta)\rceil}$ level points of the annulus. We patch these edges via the MST of $H_{i-\lceil\log(1/\delta)\rceil}$ level points in S_1 (and these points are also found in S_2). Since there are $(d/\varepsilon)^{O(d)}$ such points, the MST has weight $(d/\varepsilon)^{O(d)}2^i$, which is less than $\varepsilon q 2^i < \varepsilon w(\mathrm{MST}(S_2))$, for an appropriate choice of q.

Now consider the smaller edges of length less than $\frac{\varepsilon}{\delta}2^i$ cut by the split. These edges must all be incident on points inside the annulus, so they can all be patched by the annulus MST of weight $\varepsilon q 2^i$.

(iii): The overlap of S_1 and S_2 and their hierarchies amount to $(d/\varepsilon)^{O(d)}$ points. The MST of the overlap points is at most $(d/\varepsilon)^{O(d)}2^i$ which is less than $\mathrm{MST}(S_2)$ for an appropriate choice of q. Hence, S_2 must contain points not found in S_1, and S_1 is a proper subset of S.

(iv): Since $\tilde{\mathcal{H}}$ possesses b-neighbor links, the entire procedure described above can all be done in time $(d/\varepsilon)^{O(d)}|S_1|$.

It remains only to demonstrate that a relatively light weight annulus must exist. Note first that as a consequence of Lemma V.2 applied to v and any radius s, the sum of the weights of all distinct annuli in $B^\delta(v,s)$ is only a constant fraction times $w(B^\delta(v,s))$. Now, if we find any annulus whose weight is not more than a factor $g\varepsilon$ (for some constant g) times the sum of the weights of all annuli internal to the current annulus, then we may take that annulus and it satisfies the conditions of the lemma (up to constant factors). It must be that some annulus satisfies this condition, for if no annuli satisfy this condition, then the weight of the sum of all annuli internal to $B^\delta(v, 4\cdot 2^i)$ is at least $(1+g\varepsilon)^{d/\varepsilon} = \Theta(2^{gd})$ times $w(B^\delta(v,2^i))$, and then Lemma V.2 implies that the weight of $w(B^\delta(v, 2\cdot 2^i))$ must be $\Omega(2^{gd})$ times $w(B^\delta(u,2^i))$. For appropriate choice of g, this exceeds the weight upper bound implied by the covering property (as in **(i)** above). ∎

Having presented Lemma V.3, we can now proceed to prove Theorem V.1.

Proof of Theorem V.1: Our plan is to use Lemma V.3 to repeatedly remove from S subsets which approach a density upper bound. We then show that each subset can be broken off of S while increasing the total tour cost and total set sizes by only a small amount. This increase will be charged to the removed set.

We begin by incrementally building the graph G for S, executing the algorithm of Section IV. As described, the algorithm begins at the bottom hierarchical level H_0 and proceeds upwards. At each level H_i, it builds partial graph G_i by adding the appropriate H_i level edges to G_{i-1}. During the run of the algorithm, we can simultaneously maintain the weight of each point and subtree rooted in H_i. If we see that no point in level H_i possesses a subtree of weight greater than qs^i, then by Lemma IV.1 and the packing property of Euclidean spaces, we conclude that all balls $B(u, O(2^i))$ have MST weight bounded by $2^{O(d)}q2^i$. Otherwise, if during the construction process we encounter a level H_i in which one or more points v possesses a subtree of weight greater than $q2^i$, we invoke Lemma V.3 to identify a set S_2 which can be split from S, leaving set S_1. For S_1, we repair the compressed hierarchy $\tilde{\mathcal{H}}$ and graph G of S, all in $O_d(|S_2|)$ time. Note that the repair to G may change the weight of other subtrees in levels H_j $j \le i$, but this is not problematic: Although subtrees in H_j may increase in weight, all subtrees rooted at H_j were already determined to be sparse, and removal of points from S cannot increase the local MST. Once all dense sets in H_i are removed, we proceed to level H_{i+1}. At the conclusion of this process, we have achieved the sparsity bound of the theorem.

It remains to show that the subtree weights can be maintained efficiently under deletions. This follows from the fact that every H_j level edge added to $\tilde{\mathcal{H}}$ (for $j \le i$) due to the deletion of $v \in H_0$ has an endpoint with distance $\sum_{k=0}^{j} 2c\cdot 2^k < 4c\cdot 2^j$ of v, so this endpoint is within distance $4c\cdot 2^j + 2\cdot 2^j = (4c+2)2^j$ of v's ancestor in H_j. It follows that all affected points of G_i are $(4c+2)$-neighbors of an ancestor of v in the hierarchy $\tilde{\mathcal{H}}_2$. Assuming the hierarchy maintains b-neighbors for $b \ge 4c+2 = 26$, all updates to the affected points can be charged to hierarchical points in $\tilde{\mathcal{H}}_2$, and $|\tilde{\mathcal{H}}_2| = O(|S_2|)$.

The stated runtime follows from recalling that G can be constructed in linear time, and noting that the cost of removing a subset using Lemma V.3 is linear in the size of subset or hierarchy points being removed. The claimed tour cost and set size follow from charging the tour and size increase in Lemma V.3 to the tours and points of each removed subset S_2. ∎

VI. TSP ALGORITHM FOR SPARSE SETS

In Section V, we showed how to split the set S into subsets each with a sparse (or light-weight) MST. In this section, we show that sparse sets admit fast algorithms for finding almost optimal tours. Recall that the greedy hierarchical spanner is one built by the execution of the greedy algorithm on the complete hierarchical spanner. We will prove the following theorem:

Lemma VI.1. *Given a point set S which is $2^{O(d)}q2^i$-sparse for every ball, along with the compressed hierarchy $\tilde{\mathcal{H}}$ of S; In time $2^{O(d)}|S|$ we can compute the greedy hierarchical spanner of S and $\tilde{\mathcal{H}}$, along with a hierarchical clustering wherein each H_i level cluster cuts at most $r = (d/\varepsilon)^{O(d)}$ edges of the spanner over all levels H_j, $j \le i$.*

Before proving the lemma, we note that given the clustering of Lemma VI.1, the standard dynamic programming algorithm functioning on the clustering yields an approximate tour for the point set. This fact is expressed in the following corollary:

Corollary VI.2. *Given a point set S which is $2^{O(d)}q2^i$-sparse for every ball, and the compressed hierarchy $\tilde{\mathcal{H}}$ of S; in time $2^{O(r)}|S| = 2^{(d/\varepsilon)^{O(d)}}|S|$ we can compute a $(1+\varepsilon)$-approximate tour for S.*

Finally, Theorem I.1 follows immediately from Corollary VI.2 and the sparse decomposition of Theorem V.1. It remains only to prove Lemma VI.1, which we do in the next section.

A. Sparse clustering

In this section we address the proof of Lemma VI.1. Consider the following standard construction: For each level H_i, we assign an arbitrary ordering to the points, and choose for each point a different random radius in the range $[2^i, 2\cdot 2^i]$. We then create a partition for level H_i, assigning to each $u \in H_i$ all previously unassigned points within the radius of u. A clustering can then be constructed, for example, by building the the bottom level partition, and allowing the higher levels to segment the lower partition into smaller clusters. (This construction differs from that of [FRT03] in that each center of each level is assigned a different random radius.)

Note that the above clustering already has some favorable properties: The probability that an arbitrary ball of level H_j ($j < i$) is cut by an H_i partition in this scheme is $2^{O(d)}2^j/2^i$ (this follows easily from [Bar96], [GKL03]), and so in expectation an $H_j \in \tilde{\mathcal{H}}$ ball to be cut by only $O(d)$ higher level partitions – that is, the probability of the H_j ball to be cut decreases exponentially once its size is smaller than the larger balls by a factor $2^{-O(d)}$. By linearity of expectation, this implies that the entire random clustering can be constructed in expected linear time: The construction time for each partition is proportional to the number of balls that partition cuts, so the total runtime is the sum of the number of times each ball is cut.

Suppose now that the edges of the greedy hierarchical spanner were known. A similar argument as above can be applied to all H_j level edges of the greedy spanner: Since each edge has length $O(2^j/\varepsilon)$, we expect each one to be cut by only $O(d+\log(1/\varepsilon))$ higher level partitions. By Lemma V.3(i), all balls have locally sparse MST, and the local weight of the greedy spanner is bound by the local weight of the MST [DN97]. So a ball $B(u,R)$ covers edge weight $q2^{O(d)}R$, and therefore each cluster cuts an expected number $q2^{O(d)} = (d/\varepsilon)^{O(d)}$ edges. Hence, a random clustering possesses the necessary construction time and lightness properties *in expectation*; we must show how to achieve these bounds with high probability while simultaneously constructing the greedy hierarchical spanner.

The procedure begins at level H_0: We give each point $u \in H_0$ a different random radius in the range $[1,2]$: The cluster for u is assigned all previously unassigned points within the radius of u. We simultaneously build the bottom level of the greedy hierarchical spanner by enumerating all interpoint distances in the range $[1, c/\varepsilon]$, sorting them into $O(1/\varepsilon)$ buckets of additive increment ε. We then running the greedy algorithm by determining for each pair whether a low-stretch path already exists – it suffices for each point to inspect edges connecting $\varepsilon^{-O(d)}$ neighbors.

The procedure then considers each higher level H_i separately. Assume by induction that the clustering and the greedy spanner edges of levels H_j $j < i$ have already been constructed. An arbitrary ordering is placed on the points, after which we assign each point a random radius. We want a radius which cuts $(d/\varepsilon)^{O(d)}$ greedy spanner edges over all levels H_j $j < i$, and as above a constant fraction of radii satisfy this requirement. We further require that the radius not cut 'too many' balls of $\tilde{\mathcal{H}}$ – that is, at most a constant factor more than the average over all radii for this center. This average value can be estimated by storing at each point the sum of the radii of the nodes in the point's subtree. (Recall that probability that the ball centered at the node is cut, is linearly proportional to the node's radius.) Given the random radius, we test its validity by inspecting the balls and edges of levels H_j for $j = i, i-1, \ldots$ cut by this radius. If we discover that it cuts a constant factor more balls or edges than the estimated average, we sample a new radius and retest. For a given center, we will allow $n^{1/3d}$ resamples before declaring that the algorithm has failed.

Once we have discovered valid radii for all H_i centers, we use the radii to create H_i level clusters in the usual way. There is a single caveat: If an H_i level partition cuts a lower level cluster into two, such that one of the halves now has too many edges incident upon it, then we either include all or none of the lower level cluster as part of the larger one. More specifically, suppose cluster B is cut by an H_i level partition into two distinct cluster B_1, B_2, B_1 inside the partition and B_2 outside of it. B had previously cut only b edges, b_1 of which are now incident on B_1 and b_2 of which are now incident on B_2 (where $b_1 + b_2 = b$). The cut also introduce b' new cut edges incident on both B_1 and B_2. Now, suppose one of $b_1 + b'$ or $b_2 + b'$ exceeds the allowed number of cuts: If $b' > b_2$, then we incorporate the entire cluster B into the partition, and this can only reduce the number of edges incident on the partition. Otherwise it must be that $b' > b_1$, in which case we eject the entire cluster B from the partition, which again decreases the number of edges incident on the partition.

After creating a cluster, we calculate the spanner distances from the cluster center to its $(d/\varepsilon)^{O(d)}$ H_i level exit portals: For every newly cut edge, beginning at the lowest level cluster in which it is cut, we calculate the spanner distance from the edge to all other edges exiting its cluster. Having computed the distance from the edge to all edges exiting the H_j level cluster, the distance from the edge to other edges exiting H_{j+1} level clusters can be computed in time $(d/\varepsilon)^{O(d)}$, so that the total runtime is proportional to

$(d/\varepsilon)^{O(d)}$ times the number of lower level balls cut by H_i level partition. Given the distances from cluster centers to the cut edges, we can calculate the distance from cluster centers on the partial greedy spanner, and determine what H_i level edges must be added to the spanner in $(d/\varepsilon)^{O(d)}$ time per H_i level point.

It remains to show that the algorithm succeeds in linear time with high probability $1 - e^{-O_d(n^{1/3d})}$. The clustering algorithm fails if (i) any radius is resampled $n^{1/3d}$ times, or (ii) the total runtime exceeds some fixed constant times n. The first event occurs with probability $2^{-O(n^{1/3d})}$. In the remainder of this section we will bound the later event.

Let X_i be the maximum allowed runtime for a single radii test of point $p_i \in H_j$, and recall that X_i is proportional to the expected number of smaller balls cut by a random ball $B(p_i, O(2^i))$. As mentioned above, $\sum_i X_i = O_d(|\tilde{\mathcal{H}}|)$. We can also show that for any p_i, $X_i = O_d(|\tilde{\mathcal{H}}|^{1-1/d})$: Clearly, the worst-case is one where the all $|\tilde{\mathcal{H}}|-1$ hierarchical balls are found in $B(p_i, O(2^j))$, and further that the balls are in the highest level possible (since a ball with a larger radius is more likely to be cut). Recall that the probability that a ball of level H_k is cut by an H_j ball ($j > k$) is $2^{O(d)}2^k/2^j$, and so X_i is bound by $2^{O(d)} \sum_{f=1}^{(\log|\tilde{\mathcal{H}}|)/d} \frac{2^{df}}{2^i} = O_d(|\tilde{\mathcal{H}}|^{1-1/d})$, where the numerator represents the maximum number of balls which can be packed in the k-th level below the point, and the numerator is the ratio between the point's radius and the radius of the k-th level below.

Let random variable Z_i be the total time taken to determine a radius for p_i, that is the combined runtime for all the radii tests for p_i until a valid radius is found. The total runtime of the algorithm is $\sum_i Z_i$, and since the probability of requiring k samples is only $2^{-O(k)}$, we have $\mathbb{E}[\sum_i Z_i] = \sum_i \mathbb{E}[Z_i] = \sum_i O(X_i) = O_d(|\tilde{\mathcal{H}}|)$. Since at most $n^{1/3d}$ resamples are permitted, the maximum value attained by Z_i is bounded by $n^{1/3d}X_i = O_d(n^{1/3d}|\tilde{\mathcal{H}}|^{1-1/d})$.

By Hoeffding's inequality, the probability that the total runtime $\sum_i Z_i$ exceeds its expectation by more than $|\tilde{\mathcal{H}}|$ is $\Pr[\sum_i Z_i - E[\sum_i Z_i] \geq |\tilde{\mathcal{H}}|] = e^{-O_d(|\tilde{\mathcal{H}}|^2/\sum_i (n^{1/3d}X_i)^2)} = e^{-O_d(|\tilde{\mathcal{H}}|^2/n^{2/3d}\sum_i (X_i)^2)} = e^{-O_d(|\tilde{\mathcal{H}}|^2/n^{2/3d}|\tilde{\mathcal{H}}|^{1-1/d}\sum_i X_i)} = e^{-O_d(|\tilde{\mathcal{H}}|^2/n^{2/3d}|\tilde{\mathcal{H}}|^{1-1/d}|\tilde{\mathcal{H}}|)} = e^{-O_d(|\tilde{\mathcal{H}}|^{1/d}/n^{2/3d})} = e^{-O_d(n^{1/3d})}$.

Acknowledgements: We thank Neal Young for references, and Timothy Chan for very helpful discussions.

REFERENCES

[ABCC07] David L. Applegate, Robert E. Bixby, Vasek Chvatal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA, 2007.

[AMS99] S. Arya, D. M. Mount, and M. Smid. Dynamic algorithms for geometric spanners of small diameter: Randomized solutions. *Computational Geometry: Theory and Applications*, 13:91–107, 1999.

[Aro98] Sanjeev Arora. Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM*, 45:753–782, 1998.

[ARR98] Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *30th annual ACM symposium on Theory of computing*, pages 106–113. ACM, 1998.

[Bar96] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *37th Annual Symposium on Foundations of Computer Science*, pages 184–193. IEEE Computer Society, 1996.

[BE12] Glencora Borradaile and David Eppstein. Near-linear-time deterministic plane steiner spanners and TSP approximation for well-spaced point sets. In *CCCG*, pages 297–302, 2012.

[BGK12] Yair Bartal, Lee-Ad Gottlieb, and Robert Krauthgamer. The traveling salesman problem: low-dimensionality implies a polynomial time approximation scheme. In *Proceedings of the 44th symposium on Theory of Computing*, STOC '12, pages 663–672, 2012.

[BGM04] P. Bose, J. Gudmundsson, and P. Morin. Ordered theta graphs. *Computational Geometry: Theory and Applications*, 28:11–18, 2004.

[CE11] T.-H. Hubert Chan and Khaled M. Elbassioni. A QPTAS for TSP with fat weakly disjoint neighborhoods in doubling metrics. *Discrete & Computational Geometry*, 46(4):704–723, 2011.

[CG06] Richard Cole and Lee-Ad Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *38th annual ACM symposium on Theory of computing*, pages 574–583, 2006.

[CG08] T-H. Hubert Chan and Anupam Gupta. Approximating TSP on metrics with bounded global growth. In *19th annual ACM-SIAM symposium on Discrete algorithms*, pages 690–699. SIAM, 2008.

[Cha08] Timothy M. Chan. Well-separated pair decomposition in linear time? *Information Processing Letters*, 107(5):148–141, 108.

[CK95] P. B. Callahan and S.R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. ACM*, 42:67–90, 1995.

[CL98] Artur Czumaj and Andrzej Lingas. A polynomial time approximation scheme for euclidean minimum cost k-connectivity. In *25th International Colloquium on Automata, Languages and Programming*, ICALP '98, pages 682–694. Springer-Verlag, 1998.

[CLZ02] Artur Czumaj, Andrzej Lingas, and Hairong Zhao. Polynomial-time approximation schemes for the euclidean survivable network design problem. In *29th International Colloquium on Automata, Languages and Programming*, ICALP '02, pages 973–984. Springer-Verlag, 2002.

[CRT05] Bernard Chazelle, Ronitt Rubinfeld, and Luca Trevisan. Approximating the minimum spanning tree weight in sublinear time. *SIAM J. Comput.*, 34(6):1370–1379, 2005.

[DKS97] G. Das, S. Kapoor, and M. Smid. On the complexity of approximating euclidean traveling salesman tours and minimum spanning trees. *Algorithmica*, 19(4):447–462, 1997.

[DN97] Gautam Das and Giri Narasimhan. A fast algorithm for constructing sparse euclidean spanners. *Int. J. Comput. Geometry Appl.*, 7(4):297–315, 1997.

[DRK95] Robert Endre Tarjan David R. Karger, Philip N. Klein. A randomized linear-time algorithm to find minimum spanning trees. *Journal of the ACM*, 42:321–328, 1995.

[Epp12] David Eppstein. Steiner spanners. `http://11011110.livejournal.com/249479.html`, June 2012.

[FRT03] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *35th annual ACM symposium on Theory of computing*, pages 448–455. ACM, 2003.

[FW94] Michael L. Fredman and Dan E. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *J. Comput. Syst. Sci.*, 48(3):533–551, June 1994.

[Gar82] Irene Gargantini. An effective way to represent quadtrees. *Commun. ACM*, 25(12):905–910, December 1982.

[GGN06] Jie Gao, Leonidas J. Guibas, and An Nguyen. Deformable spanners and applications. *Comput. Geom. Theory Appl.*, 35(1), 2006.

[GKL03] Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *44th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '03, pages 534–543. IEEE Computer Society, 2003.

[GLN02] Joachim Gudmundsson, Christos Levcopoulos, and Giri Narasimhan. Fast greedy algorithms for constructing sparse geometric spanners. *SIAM Journal on Computing*, 31:2002, 2002.

[GP02] Gregory Gutin and Abraham P. Punnen, editors. *The traveling salesman problem and its variations*, volume 12 of *Combinatorial Optimization*. Kluwer Academic Publishers, Dordrecht, 2002.

[Har10] Sariel Har-Peled. Data-structures for geometric approximation. Manuscript, 2010.

[KL04] Robert Krauthgamer and James R. Lee. Navigating nets: Simple algorithms for proximity search. In *15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 791–801, January 2004.

[Kle08] P. Klein. A linear-time approximation scheme for TSP in undirected planar graphs with edge-weights. *SIAM Journal on Computing*, 37(6):1926–1952, 2008.

[KR07] Stavros G. Kolliopoulos and Satish Rao. A nearly linear-time approximation scheme for the Euclidean k-median problem. *SIAM J. Comput.*, 37:757–782, June 2007.

[LLKS85] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem*. Wiley-Interscience series in discrete mathematics, 1985.

[Mit99] Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999.

[Mit07] Joseph S. B. Mitchell. A PTAS for TSP with neighborhoods among fat regions in the plane. In *8th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 11–18. SIAM, 2007.

[Rei94] G. Reinelt. *The Traveling Salesman*, volume 840 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1994.

[RS98] Satish B. Rao and Warren D. Smith. Approximating geometrical graphs via "spanners" and "banyans". In *30th annual ACM symposium on Theory of computing*, pages 540–550. ACM, 1998.

[Sal91] J. S. Salowe. Constructing multidimensional spanner graphs. *Int. J. Comput. Geometry Appl*, 1(2):99–107, 1991.

[Soa94] J. Soares. Approximating euclidean distances by small degree graphs. *Discrete & Computational Geometry*, 11:213–233, 1994.

[Tal04] Kunal Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *36th annual ACM symposium on Theory of computing*, pages 281–290. ACM, 2004.

[Tre00] L. Trevisan. When Hamming meets Euclid: The approximability of geometric TSP and Steiner tree. *SIAM Journal on Computing*, 30(2):475–485, 2000.

[Vai91] P. M. Vaidya. A sparse graph almost as good as the complete graph on points in K dimensions. *Discrete & Computational Geometry*, 6:369–381, 1991.

[You13] Neal Young. Approximating 1-dimensional TSP requires $\Omega(n \log n)$ comparisons. Manuscript, available at `http://arxiv.org/abs/1303.2920`, 2013.