Time complexity of the Analyst's Traveling Salesman algorithm

ANTHONY RAMIREZ

VYRON VELLIS

Abstract: The Analyst's Traveling Salesman Problem asks for conditions under which a (finite or infinite) subset of $\mathbb{R}^N$ is contained on a curve of finite length. We show that for finite sets, the algorithm constructed in Schul [22] and Badger, Naples and Vellis [3] that solves the Analyst's Traveling Salesman Problem has polynomial time complexity and we determine the sharp exponent.

1 Introduction

The *(Euclidean) Traveling Salesman Problem (TSP)* (Lawler, Lenstra and Rinnooy Kan [16], Gutin and Punnen [12], and Applegate, Bixby, Chvátaland and Cook [1]) asks to find the shortest path through a set V of n points in $\mathbb{R}^N$ that starts and ends at a given vertex v_0 of V. Apart from its natural applications in itinerary design, and its influence on operations research, and polyhedral theory, in the last 50 years the TSP has gained great prominence in computer science due to its immense computational complexity. For example, it is well known that the TSP is NP–hard, see Garey, Graham, and Johnson [8] and Papadimitriou [20]; that is, it is at least as difficult as the hardest problems in NP (the class of all problems that can be solved by a non-deterministic Turing machine in polynomial time $O(n^k)$ for some $k \in \mathbb{N}$).

What is the minimum amount of time (depending on n) which is required to obtain a solution of the TSP? Can it be polynomial (ie $O(n^k)$ for some $k \in \mathbb{N}$)? Obviously, one could simply try all possible paths but that would require time comparable to $(n-1)!$ which is far from being polynomial. The Bellman–Held–Karp algorithm [6, 13] improves the latter bound to $O(n^2 2^n)$. However, it is still unknown whether a polynomially complex algorithm exists.

In lieu of the above discussion, "nearly-optimal" algorithms have been explored. That is, algorithms that produce a path which may not be the optimal one but is comparable (or even arbitrarily close) in length to the optimal one. The *nearest insertion algorithm* (Rosenkrantz, Stearns and Lewis [21]) computes in $O(n^2)$ a path which is at most twice in length of the optimal one; see also Golden, Bodin, Doyle and Stewart [9] and Lin and Kernighan [17] for heuristics of similar time complexity. Christofides heuristic [7] computes in $O(n^3)$ time a path that is at most $3/2$ times the length of the optimal one. Grigni, Koutsoupias and Papadimitriou [10] designed an algorithm for planar graphs that for all $\epsilon > 0$ provides in $O\left(n^{O(1/\epsilon)}\right)$ time a tour of length at most $(1 + \epsilon)$ times the optimal one. A similar result of $O(n^{O(1/\epsilon)})$ time was also obtained later by Mitchel [18]. In his celebrated work, Arora [2] constructed an algorithm which, for each $\epsilon > 0$, gives with probability more than $1/2$ a path which is at most $(1 + \epsilon)$ times the length of the optimal one in $O\left(n(\log n)^{(O(\sqrt{N}/\epsilon))^{N-1}}\right)$ time. Arora's algorithm can be derandomized by increasing the running time by $O(n^N)$. See also Bartal, Gottlieb and Krauthgamer [5] for relevant results.

The *Analyst's Traveling Salesman Problem (ATSP)* (Jones [15]) is a generalization of the TSP where it is asked to find a curve of finite length that contains a given (finite or infinite) set $V \subset \mathbb{R}^N$. While the TSP (which is the finite case of ATSP) always admits a solution, the same is not true in general for the ATSP. For instance, if V is an unbounded set, then clearly every curve that contains V must be unbounded itself, hence with infinite length. Perhaps less intuitively, smaller, but still infinite, sets may not admit a solution. For example, if V is the set of all points in the unit square $[0, 1]^2$ with rational coordinates, it is not hard to see that V is bounded and countable but there exists no rectifiable curve that contains V.

The classification of sets in $\mathbb{R}^N$ for which the ATSP can be solved, was done by Jones [15] (for $\mathbb{R}^2$) and by Okikiolu [19] (for higher dimensional spaces). Ever since, this classification has played a crucial role in the development of geometric measure theory, and has found applications in many facets of analysis including complex analysis, dynamics, harmonic analysis and metric geometry. The work of Jones and Okikiolu provides an algorithm that, for any finite set $V \subset \mathbb{R}^N$ yields a tour that is at most $C(N)$ times longer than the optimal path. Here $C(N)$ is a constant that depends only on the dimension N. The Jones–Okikiolu algorithm is based on a local version of the Farthest Insertion algorithm (Johnson and McGeoch [14]).

Later, Schul [22] provided a modification of the algorithm so that the ratio of the length of the yielded path over the length of the optimal path is bounded by a constant C independent of the dimension N. Variation of this algorithm also appears in the work of Badgers, Naples and Vellis [3]. Here and for the rest of this paper, we refer to any of

these two variations as the ATSP algorithm. The purpose of this note is to show that the ATSP algorithm, in the case that V is finite, has polynomial time complexity.

Theorem 1.1 *The time complexity of the ATSP algorithm is $O(n^3)$.*

We remark that although time-wise this algorithm is fast, the ratio constant of the yielded path over the optimal path has not been computed and is much larger that Christofides' $3/2$ ratio. In fact, in our algorithm, the yielded path has length at most $(300)^{9/2} \log 300$ times the length of the smallest spanning tree. Moreover, the exponent 3 in our theorem is sharp and can not be lowered; see Section 5.4.

An interesting connection to the Jones-Scul algorithm was given by Gu, Lutz and Mayordomo [11] who classified the sets V that admit a solution to a computable extension of the ASTP. This variant of the problem characterizes the sets which are contained in a rectifiable computable curve. As we are concerned only with finite sets here, our algorithm already produces computable curves.

1.1   Outline of the ATSP algorithm

Fix a set $V = \{v_1, \ldots, v_n\} \subset \mathbb{R}^N$. The algorithm (described in detail in Section 5) computes connected graphs $G_k = (V_k, E_k)$ with $k = 1, \ldots, m$ for some $m \leq n$, so that the last graph satisfies $V_m = V$. First, we compute $R_0 = 5 \max\{|v| : v \in V\}$ which takes $O(n)$ time. It is clear that $V \subset [-R_0, R_0]^N$. The construction of graphs now is roughly as follows.

Step 1**:**
For the first graph we have $V_1 = \{v_1\}$ and $E_1 = \emptyset$. We set $n_1 = 1$ and $U_1 = V \setminus V_1$. If $U_1 = \emptyset$, then proceed to the *Final Step*. Otherwise proceed to the next step.

Step $k + 1$**:**
Inductively, we assume that for some $k \in \mathbb{N}$, we have defined an integer $n_k \in \mathbb{N}$, two sets $V_k, U_k \subset V$, and a graph $G_k = (V_k, E_k)$ such that $U_k = V \setminus V_k$, $U_k \neq \emptyset$, and V_k is a maximal $(2^{-n_k} R_0)$–separated set. We use results in Section 3 to define an integer $n_{k+1} > n_k$, a set V_{k+1} that contains V_k and is contained in V, and a set $U_{k+1} = V \setminus V_{k+1}$ such that V_{k+1} is a maximal $(2^{-n_{k+1}} R_0)$–separated set and $\mathrm{dist}_H(V_k, V_{k+1}) < 2^{-n_{k+1}} R_0$. Here and for the rest of this paper, $\mathrm{dist}_H(A, B)$ denotes the Hausdorff distance between closed sets $A, B \subset \mathbb{R}^N$:

$$\mathrm{dist}_H(A, B) = \max \left\{ \sup_{x \in A} \inf_{y \in B} |x - y|, \ \sup_{y \in B} \inf_{x \in A} |x - y| \right\}$$

Next, using results in Section 4, we define for each $v \in V_k$ a number α that measures how "flat" the set V_{k+1} is around the point v. This notion of flatness is inspired by the *Jones beta numbers* in the work of Jones [15] and Schul [22].

Next, following the constructions in Schul [22] and Badger, Naples and Vellis [3], we create a connected graph $G_{k+1} = (V_{k+1}, E_{k+1})$. Roughly speaking, if around some node v of G_k, the set V_{k+1} looks flat (ie the number α is small), then the graph G_{k+1} around v should also be flat; otherwise, we add new edges around v so that the new graph is connected. We take care so that the total number of new edges is at most twice the total number of new points.

Final Step:

Arriving at this step, we have created a graph $G_m = (V_m, E_m)$ with $V_m = V$. Since each V_k is different than V_{k-1}, we arrive at the final step in at most n steps. We show that each step above takes $O(n^2)$ time, and therefore, we obtain G_m in $O(n^3)$ time. Now we use an algorithm in Section 2.2 to parameterize G_m in $O(n^3)$ time.

1.2   Hölder curves

It should be noted that the work of Jones [15], Okikiolu [19], and Schul [22] provides the sets for which a solution exists but not the solution itself when the set is infinite. That is, they classify the sets which are contained in curves of finite length, but do not provide the parametrization of the curves. Recently, the second named author with Badger and Naples [3] constructed an algorithm that, for those sets V that have a solution in ATSP, produces a solution path $f : [0, 1] \to \mathbb{R}^N$ which is at most C times longer than the optimal path with C independent of N. In this algorithm, one obtains parameterizations (f_k) for *all graphs* G_k and not just one of them. Moreover, the parameterizations obtained satisfy the inequality $\|f_k - f_{k+1}\|_\infty \leq 2^{-n_{k+1}}$ for all k. Although this approach is more complicated than the one of Jones and Okikiolu, it comes very handy in designing Hölder parameterizations of sets such as the space-filling Peano curve. For finite sets, the algorithm of constructing (f_k) has also polynomial complexity (and in fact with the same exponent) but we will not pursue this direction here.

2   Preliminaries on graphs

A *(combinatorial) graph* is a pair $G = (V, E)$ of a finite vertex set V and an edge set:
$$E \subset \{\{v, v'\} : v, v' \in V \text{ and } v \neq v'\}$$

A graph $G' = (V', E')$ is a *subgraph* of $G = (V, E)$ (and we write $G \subset G'$) if $V' \subset V$ and $E' \subset E$. A *path* in G is a set $\gamma = \{\{v_1, v_2\}, \ldots, \{v_{n-1}, v_n\}\} \subset E$; in this case we say that γ joins v_1, v_n. A combinatorial graph $G = (V, E)$ is connected, if for any distinct $v, v' \in V$ there exists a path γ in G that joins v with v'. A *connected component* of a combinatorial graph G is a maximal connected subgraph of G.

2.1 Components of a graph

Given a graph $G = (V, E)$ we describe an algorithm that returns the connected components of G.

Lemma 2.1 *There exists an algorithm that determines the components of G in $O\left((\operatorname{card} V)(\operatorname{card} E)^2\right)$ time.*

Proof We fix $v_0 \in V$ and let $(V_{1,1}, E_{1,1}) = (\{v_0\}, \emptyset)$. Assuming that for some $i \leq \operatorname{card} E - 1$ we have defined $(V_{1,i}, E_{1,i})$, there are two possible cases.

(1) If there exists $e \in E \setminus E_{1,i}$ such that e has an endpoint in $V_{1,i}$, then we set $V_{1,i+1} = V_{1,i} \cup e$ and $E_{1,i+1} = E_{1,i} \cup \{e\}$.

(2) If there does not exist $e \in E \setminus E_{1,i}$ such that e has an endpoint in $V_{1,i}$, then we stop the procedure and set $V_1 = V_{1,i}$ and $E_1 = E_{1,i}$.

For each i, we make sure in $O(\operatorname{card} V_1)$ that no vertex appears twice. The time needed for the calculation of V_1 is at most a constant multiple of:

$$\sum_{i=1}^{\operatorname{card} E_1} (\operatorname{card} V_{1,i}) \operatorname{card} E = O\left((\operatorname{card} V_1)(\operatorname{card} E)^2\right)$$

If $V_1 \neq V$, we replace (V, E) by $(V \setminus V_1, E \setminus E_1)$ and we repeat the process again to obtain (V_2, E_2). We continue this way until we exhaust all points of V. Since the V_i's are disjoint, the sum of their cardinalities is $\operatorname{card} V$. Thus, the total time needed for the algorithm to complete is at most a constant multiple of:

$$O((\operatorname{card} V_1)(\operatorname{card} E)^2) + O((\operatorname{card} V_2)(\operatorname{card} E)^2) + \cdots = O((\operatorname{card} V)(\operatorname{card} E)^2) \quad \square$$

2.2 Two-to-one tours on connected graphs

Given a connected graph $G = (V, E)$, we describe here an algorithm that gives a tour over all edges of G so that each edge is traversed exactly twice and once in each direction.

Lemma 2.2 *Let $G = (V, E)$ be a finite connected graph and let $v_0 \in V$. There exists an algorithm which in $O((\operatorname{card} V)(\operatorname{card} E)^2)$ time produces a finite sequence $(a_i)_{i=1}^{2M+1}$ of points in V with $M = \operatorname{card} E$ that satisfies the following properties.*

(1) *We have $\{a_1, \ldots, a_{2M+1}\} = V$ with $a_1 = a_{2M+1} = v_0$.*
(2) *For any $i \in \{1, \ldots, 2M\}$, $\{a_i, a_{i+1}\} \in E$.*
(3) *For any $e \in E$ there exist exactly two distinct $i, j \in \{1, \ldots, 2M\}$ such that*

$$\{a_i, a_{i+1}\} = \{a_j, a_{j+1}\} = e.$$

Moreover, $a_i = a_{j+1}$ and $a_j = a_{i+1}$.

Proof Set $a_{0,1} = v_0$ and $E_0 = E$.

Assume that for some odd $k \in \{0, 1, \ldots, M-1\}$ we have defined a set $E_k \subset E$ and a finite sequence $(a_{k,i})_{i=1}^{2k+1}$ with the following properties.

(1) The set $\{a_{k,1}, \ldots, a_{k,2k+1}\} \subset V$ with $a_{k,1} = a_{k,2k+1} = v_0$.
(2) For any $i \in \{1, \ldots, 2k\}$, $\{a_{k,i}, a_{k,i+1}\} \in E \setminus E_k$.
(3) For any $e \in E \setminus E_k$ there exist exactly two distinct $i, j \in \{1, \ldots, n-1\}$ such that

$$\{a_{k,i}, a_{k,i+1}\} = \{a_{k,j}, a_{k,j+1}\} = e.$$

Moreover, $a_i = a_{j+1}$ and $a_j = a_{i+1}$.

Find $e \in E_k$ and $j \in \{0, 1, \ldots, 2k+1\}$ such that $a_{k,j} \in e$. Since G is assumed connected, such j and e exist and the search for them would take at most $O(\operatorname{card} E \operatorname{card} V)$ time. Let v be the unique element of $e \setminus \{a_{k,j}\}$. Define now $E_{k+1} = E_k \setminus \{e\}$ and:

$$a_{k+1,i} = \begin{cases} a_{k,i} & \text{if } 1 \leq k \leq j \\ v & \text{if } i = j \\ a_{k,i-2} & \text{if } j+2 \leq i \leq 2k+3 \end{cases}$$

The construction of E_{k+1} and the sequence $(a_{k+1,i})_{i=1}^{2k+3}$ takes $O(\operatorname{card} E)$ time. By design and the inductive hypothesis, each edge in $E \setminus E_{k+1}$ is traversed exactly twice. In particular, the properties (1)–(3) above are true for $k+1$.

This process terminates when $k = M$. The total time required is a constant multiple of:

$$\operatorname{card} E \left(O(\operatorname{card} E) + O(\operatorname{card} E \operatorname{card} V) \right) = O(\operatorname{card} V (\operatorname{card} E)^2)$$

For $i = 1, \ldots, 2M+1$ set $a_i = a_{M,i}$. Note that $E_M = \emptyset$. By induction, every edge of $E = E \setminus E_M$ is traversed twice, once in each direction. Since we traverse every edge of E, we have that $\{a_1, \ldots, a_{2M+1}\} = V$ and the proof is complete. □

3   Construction of scales and nets

Given $\epsilon > 0$, we say that a set $X \subset \mathbb{R}^N$ is ϵ–*separated* if for any $x, y \in X$ we have $|x - y| \geq \epsilon$. Given a set $V \subset \mathbb{R}^N$ and $\epsilon > 0$, we say that a set $X \subset V$ is an ϵ–*net* if it is a maximal ϵ–separated subset of V.

Recall the definition of Hausdorff distance from Section 1.1. In the next lemma we describe an algorithm that gives nets for a given set V.

Lemma 3.1   *Let $V \subset \mathbb{R}^N$ be a set of n elements, let $\epsilon > 0$, let X be an ϵ–net of V, and let $U = V \setminus X$. Assume that $U \neq \emptyset$. Then, in $O(n^2)$ time we can compute an integer $k \in \mathbb{N}$, a $(2^{-k}\epsilon)$–net X' of V, and a set $U' = V \setminus X'$ such that $X \subsetneq X' \subset V$ and:*

$$(3\text{–}1) \qquad\qquad 2^{-k}\epsilon \leq \text{dist}_H(X, X') < 2^{1-k}\epsilon$$

Proof   Assume that $X = \{x_1, \ldots, x_l\}$ and $U = \{u_1, \ldots, u_{n-l}\}$, with $l \in \{1, \ldots, n\}$, such that $U = V \setminus X$. We first calculate:

$$d = \max_{i=1,\ldots,n-l} \min_{j=1,\ldots,l} |x_j - u_i|$$

Since both sets X, U have cardinalities at most n, the computation of d takes at most $O(n^2)$ time. Let k be the smallest integer such that $2^{-k}\epsilon \leq d$; the computation of k clearly takes $O(1)$ time. By minimality of k we have that $2^{1-k}\epsilon > d$.

We initiate the construction by setting $a_i = x_i$ for $i \in \{1, \ldots, l\}$, $c_j = u_j$ for $j \in \{1, \ldots, n - l\}$. We also set $X'_1 = \{a_1, \ldots, a_l\}$, $U'_1 = \emptyset$, and $U''_1 = \{c_1, \ldots, c_{n-l}\}$. This requires $O(n)$ time.

Inductively, assume that after m steps, with $m \in \{1, \ldots, n - l\}$, we have defined three disjoint sets

$$X'_m = \{a_1, \ldots, a_p\}, \quad U'_m = \{b_1, \ldots, b_{l+m-p-1}\}, \quad U''_m = \{c_1, \ldots, c_{n-l-m+1}\}$$

such that $(X'_m \setminus X) \cup U'_m \cup U''_m = U$. Conventionally, if $l + m - p - 1 = 0$, then we assume $U'_m = \emptyset$. If

$$(3\text{–}2) \qquad\qquad \min_{i=1,\ldots,p} |c_{n-l-m+1} - a_i| \geq 2^{-k}\epsilon$$

then we set $a_{p+1} = c_{n-l-m+1}$ and:

$$X'_{m+1} = X'_m \cup \{a_{p+1}\}, \quad U'_{m+1} = U'_m, \quad U''_{m+1} = U''_m \setminus \{c_{n-l-m+1}\}$$

Otherwise, we set $b_{l+m-p} = c_{n-l-m+1}$ and:

$$X'_{m+1} = X'_m, \quad U'_{m+1} = U'_m \cup \{b_{l+m-p}\}, \quad U''_{m+1} = U''_m \setminus \{c_{n-l-m+1}\}$$

Note that the calculation of (3–2) as well as the definition of sets X'_{m+1}, U'_{m+1} and U''_{m+1} requires $O(n)$ time to complete.

Set $X' = X_{n-l+1}$ and $U' = U'_{n-l+1}$. The construction of X', U' takes $O(n^2)$ time to complete. It remains to verify the conclusions of the lemma.

By the definition of d, there exists at least one element u in U such that $|x - u| \geq 2^{-k}\epsilon$ for all $x \in X$. Therefore, the set $X' \setminus X$ is nonempty. Moreover, it is clear that X' is a $(2^{-k}\epsilon)$–separated set. To show that X' is maximal, assume for a contradiction that there exists $v \in V$ such that $|x - v| \geq 2^{-k}\epsilon$ for all $x \in X'$. Then, there exists $m \in \{1, \ldots, n - l\}$ such that $x = c_{n-l-m+1}$. But then, by (3–2), x would end up in X' which is false. The lower bound of (3–1) follows from the $2^{-k}\epsilon$–separability of X' and the fact that $X' \setminus X \neq \emptyset$. For the upper bound of (3–1) assume for a contradiction that there exists $x' \in X'$ such that $|x' - x| \geq 2^{1-k}\epsilon$ for all $x \in X$. Then, clearly $x' \in V \setminus X \subset U$ and it follows that $d \geq 2^{1-k}\epsilon$. But that contradicts the minimality of k since $k - 1$ could also work. $\qquad\square$

4 Flatness modules and flat pairs

In this section we examine a notion of local flatness introduced by Jones [15] and further developed by Schul [22] and Badger and Schul [4]. For the rest of this section we fix $C_0 = 300$, a finite set $V \subset \mathbb{R}^N$ consisting of n elements, an ϵ–net $X \subset V$, and a $(2^{-k}\epsilon)$–net $X' \subset V$ such that $X \subset X'$ and:

$$2^{-k}\epsilon \leq \text{dist}_H(X, X') < 2^{1-k}\epsilon$$

In [22] Schul defined for each $v \in X$ a number that measures the (normalized) thickness of the thinnest cylinder that contains the set:

$$B_{v,X'} := B(v, C_0 2^{-k}\epsilon) \cap X'$$

In our case, the numbers that we define measure the same thing with one difference; we will only consider a *finite* set of cylinders that contain $B_{v,X'}$ (as opposed to *all* cylinders) and we will choose the thinnest one. Nevertheless, the two flatness modules are close to each other; see Lemma 4.2.

4.1 Flatness modules

For the rest of this section, L is an integer with $L \geq 40C_0\sqrt{N-1}$ and $\mathcal{G} = \{-2L, \ldots, 2L\}^N$. Given two distinct points $y_1, y_2 \in \mathbb{R}^N$, we denote by $\ell(y_1, y_2)$

the unique line that passes through y_1, y_2 and define the normalized distance:

$$D(y_1, y_2) = \max_{z \in B_{v,X'}} 2^k \epsilon^{-1} \operatorname{dist}(z, \ell(y_1, y_2))$$

Given $v \in X$, define

$$\alpha_{v,X'} = \min_{x \in \phi(\mathcal{G})} \min_{y \in \phi(\mathcal{G}) \backslash \{x\}} D(x, y)$$

where $\phi(z) = C_0 2^{-k} \epsilon L^{-1} z + v$. Essentially, $\alpha_{v,X'}$ measures the smallest distance of the set $B_{v,X'}$ to all lines going through a fine grid of the cube $v + C_0 2^{1-k} \epsilon [-1, 1]^N$ normalized by $2^{-k} \epsilon$. We set $\ell_{v,X'}$ to be the line passing through points $x, y \in \phi(\mathcal{G})$ with $x \neq y$ that minimize the quantity $D(x, y)$.

Lemma 4.1 *Number $\alpha_{v,X'}$ and line $\ell_{v,X'}$ can be computed in $O(\operatorname{card} X')$ time.*

Proof We start by constructing the sets $B_{v,X'}$. We check over all points $u \in X'$ whether or not $|v - u| < C_0 2^{-k} \epsilon$. Then the total time needed to construct the set $B_{v,X'}$ is $O(\operatorname{card} X')$. Since the set $\mathcal{G}$ is finite and independent of n, the calculation of $\alpha_{v,X'}$ and of $\ell_{v,X'}$ takes $O(1)$ time. This completes the proof of the lemma. $\square$

Remark 4.1 Since for each $v \in X$ we have to go through the entire set X' in order to construct $B_{v,X'}$, the computations of the preceding lemma are sharp; that is, there exists universal $C > 0$ such that it takes at least $C \operatorname{card} X'$ time to compute $B_{v,X'}$.

Lemma 4.2 *If there exists a line ℓ such that $\max_{x \in B_{v,X'}} \operatorname{dist}(x, \ell) \leq (1/20)2^{-k} \epsilon$, then $\alpha_{v,X'} \leq 1/16$.*

Proof Suppose that ℓ is a line with $\max_{x \in B_{v,X'}} \operatorname{dist}(x, \ell) \leq (1/20)2^{-k} \epsilon$. Without loss of generality, we may assume that ℓ intersects the cube $v + C_0 2^{-k} \epsilon [-1, 1]^N$; otherwise we replace ℓ by a line parallel to ℓ that intersects the boundary of the cube and is closer to the set $B_{v,X'}$. Then, ℓ intersects the boundary of the cube $v + C_0 2^{1-k} \epsilon [-1, 1]^N$ on two points p_1, p_2 which have distance at least $2^{3/2} C_0 2^{-k}$. Now, we can find $p_1', p_2' \in \phi(\mathcal{G})$ such that for $i = 1, 2$:

$$|p_i - p_i'| \leq \tfrac{1}{2} \sqrt{N-1} C_0 2^{-k} \epsilon L^{-1} \leq \tfrac{1}{80} 2^{-k} \epsilon$$

By the choice of L, we have that $p_1' \neq p_2'$.

Fix $x \in B_{v,X'}$. Then, by the choice of L:

$$\operatorname{dist}(x, \ell(p_1', p_2')) \leq \operatorname{dist}(x, \ell) + \max_{i=1,2} |p_i - p_i'| \leq \tfrac{1}{16} 2^{-k} \epsilon$$

Therefore, $\alpha_{v,X'} \leq 2^k \epsilon^{-1} \max_{x \in B_{v,X'}} \operatorname{dist}(x, \ell(p_1', p_2')) \leq 1/16$. $\square$

4.2   Flat Pairs

Following Badger, Naples, and Vellis [3, Definition 2.4], for each $v \in X$ we define the
set of *flat pairs* $\mathrm{Flat}(v, X, X')$ to be the set of $\{v, v'\} \subset X$ such that

(1) $\epsilon \le |v - v'| < C_0 2^{-k-1} \epsilon$, and

(2) $\alpha_{v,X'} < 1/16$ and there exists an orientation of the line $\ell_{v,X'}$ such that v' is the
first point of $X \cap B(v, C_0 2^{-k-1}\epsilon)$ to the left or to the right of v with respect to
the given orientation.

Note that if k is large, then the first condition fails immediately and no flat pairs exist.
Here and for the rest of the paper, given a point $x \in \mathbb{R}^N$, we denote the first coordinate
of x by $\langle x \rangle_1$.

Lemma 4.3   *The set $\mathrm{Flat}(v, X, X')$ can be computed in $O(\mathrm{card}\, X')$ time.*

Proof   We first compute the set

$$B'_{v,X} := X \cap B(v, C_0 2^{-k-1}\epsilon)$$

and as in Lemma 4.1, it is easy to see that $B'_{v,X}$ can be computed in $O(\mathrm{card}\, X')$ time.
Next we compute $\alpha_{v,X'}$ which also takes $O(\mathrm{card}\, X')$ time.

If $B'_{v,X}$ contains only v, or if $\alpha_{v,X'} > 1/16$ then we set $\mathrm{Flat}(v, X, X') = \emptyset$.

Assume now that $B'_{v,X}$ contains more than v and that $\alpha_{v,X'} \le 1/16$. Having the line
$\ell_{v,X'}$ from the previous subsection, we apply an isometric affine transformation

$$\Psi : \mathbb{R}^N \to \mathbb{R}^N, \qquad \Psi(x) = A(x - v)$$

such that A is an orthogonal matrix, $\Psi(v) = 0$, and Ψ maps the line generated by
$\ell_{v,X'}$ onto the line passing trough the origin that is parallel to the x_1–axis. The matrix
A can be generated in $O(1)$ time. We compute the set $\{\langle \Psi(v') \rangle_1 : v' \in B'_{v,X}\}$ and
enumerate them from lowest to highest. By Badger and Schul [4, Lemma 8.3] no two
first coordinates can be equal.

- If there exists $v' \in B'_{v,X}$ with $\langle \Psi(v') \rangle_1 < 0$, then choose the point $v' \in B'_{v,X'}$ with
  the highest first coordinate that is negative, and place $\{v, v'\}$ in $\mathrm{Flat}(v, X, X')$.
- If there exists $v' \in B'_{v,X}$ with $\langle \Psi(v') \rangle_1 > 0$, then choose the point $v' \in B'_{v,X}$ with
  the lowest first coordinate that is positive, and place $\{v, v'\}$ in $\mathrm{Flat}(v, X, X')$.

Since $\alpha_{v,X'} \le 1/16$ and since $B'_{v,X'} \setminus \{v\}$ is nonempty, at least one of the two cases
above must hold. If both hold, then we perform both actions.

Since X is ϵ–separated, the cardinality of $B'_{v,X}$ depends only on N, and the above
computations can be done in $O(1)$ time. Hence, the computation of $\mathrm{Flat}(v, X, X')$ can
be done in $O(\mathrm{card}\, X')$ time and the proof is complete.   □

5 ATSP algorithm and proof of Theorem 1.1

Before starting the algorithm, we calculate $R_0 = 5 \max\{|v| : v \in V\}$. This takes $O(n)$ time. Here and for the rest of this section, all constants depend only on the dimension N.

5.1 Step 1:

We set $n_1 = 1$, $V_1 = \{v_1\}$, and $U_1 = V \setminus \{v_1\}$. These definitions take $O(n)$ time. If $U_1 = \emptyset$, then we move to the Final Step; see Section 5.3. Otherwise, we apply the algorithm of Lemma 3.1 for $X = V_1$ and $\epsilon = 2^{-1} R_0$ and we obtain in $O(n^2)$ time an integer $n_2 > 1$ and a $(2^{-n_2} R_0)$–net V_2 along with a set $U_2 = V \setminus V_2$. Now, we apply the algorithm of Lemma 4.1 to obtain in $O(n)$ time a number α_{v_1, V_2}. If $\alpha_{v_1, V_2} > 1/16$, then we set $\mathcal{N}_1 = \{v_1\}$ and $\mathcal{F}_1 = \emptyset$. Otherwise, we set $\mathcal{F}_1 = \{v_1\}$ and $\mathcal{N}_1 = \emptyset$. We also set $E_1 = \emptyset$ and $G_1 = (V_1, E_1)$. Step 1 takes $O(n^2)$ time.

5.2 Step $k + 1$:

Suppose that for some $k \in \mathbb{N}$ we have produced

- an integer $n_k \geq 1$,
- a $(2^{-n_k} R_0)$–net V_k and a nonempty set $U_k = V \setminus V_k$,
- two disjoint sets $\mathcal{N}_k, \mathcal{F}_k \subset V_k$ with $\mathcal{N}_k \cup \mathcal{F}_k = V_k$,
- a connected graph $G_k = (V_k, E_k)$ such that $\operatorname{card} E_k \leq 2 \operatorname{card} V_k$,
- an integer $n_{k+1} > n_k$, and
- a $(2^{-n_{k+1}} R_0)$–net V_{k+1} and a set $U_{k+1} = V \setminus V_{k+1}$.

If $U_{k+1} = \emptyset$, then this paragraph can be skipped. Otherwise, we apply the algorithm of Lemma 3.1 with $X = V_{k+1}$ and $\epsilon = 2^{-n_{k+1}} R_0$ and we obtain in $O(n^2)$ time an integer $n_{k+2} > n_{k+1}$, a $(2^{-n_{k+2}} R_0)$–net that contains V_{k+1}, and a set $U_{k+2} = V \setminus V_{k+2}$.

Next, we produce three new families $\mathcal{F}_{k+1}, \mathcal{N}_{k+1}, E_{k+1}$.

5.2.1 Families $\mathcal{N}_{k+1}$ and $\mathcal{F}_{k+1}$

We perform this part of Step $k + 1$ only if $U_{k+1} \neq \emptyset$, otherwise we omit it. For each $v \in V_{k+1}$ we compute $\alpha_{v, V_{k+2}}$ and

- if $\alpha_{v, V_{k+2}} > 1/16$, then we place v in $\mathcal{N}_{k+1}$;
- if $\alpha_{v, V_{k+2}} \leq 1/16$, then we place v in $\mathcal{F}_{k+1}$.

By Lemma 4.1, the constructions of sets $\mathcal{N}_{k+1}, \mathcal{F}_{k+1}$, takes a total of $O(n^2)$ time.

It remains to construct E_{k+1}. The new edges will come from three sources; from old edges in E_k, around points in $\mathcal{F}_k$, and around points in $\mathcal{N}_k$.

5.2.2 Edges coming from E_k

Recall from Section 4 that $C_0 = 300$. For each element $e = \{u, u'\} \in E_k$, we compute $|u - u'|$ and the two numbers $\alpha_{u, V_{k+1}}$ and $\alpha_{u', V_{k+1}}$.

If $|u - u'| \geq C_0 2^{-n_{k+1}-1} R_0$, or if both $\alpha_{u, V_{k+1}}$ and $\alpha_{u', V_{k+1}}$ are more than $1/16$, then we place e in $E_{k+1}^{(1)}$. This case takes $O(n)$ time. We set $V_{k+1}(e) = \emptyset$; this set will not play a role in the algorithm but we define it for consistency.

If $|u - u'| < C_0 2^{-n_{k+1}-1} R_0$ and at least one of $\alpha_{u, V_{k+1}}$, $\alpha_{u', V_{k+1}}$ (say $\alpha_{u, V_{k+1}}$) is less or equal to $1/16$, then we calculate $B_{u, V_{k+1}} \cup B_{u', V_{k+1}}$ and $\ell_{u, V_{k+1}}$. We also calculate an isometric affine map Ψ that takes u to the origin, takes u' to a point with positive first coordinate, and takes the line $\ell_{u, V_{k+1}}$ onto the x_1–axis. Then we determine those points $u_1, \ldots, u_l \in B_{u, V_{k+1}} \cup B_{u', V_{k+1}}$ such that the first coordinates satisfy:

$$0 = \langle \Psi(u) \rangle_1 < \langle \Psi(u_1) \rangle_1 < \cdots < \langle \Psi(u_l) \rangle_1 < \langle \Psi(u') \rangle_1$$

By Badger and Schul [4, Lemma 8.3] we know that no distinct points $\Psi(u_i)$, $\Psi(u_j)$ have equal first coordinates so the inequalities above are justified. We set $V_{k+1}(e) = \{u_1, \ldots, u_l\}$ and we add the edges $\{u, u_1\}, \ldots, \{u_l, u'\}$ in $E_{k+1}^{(1)}$. The calculation of $\alpha_{u, V_{k+1}}$ and $\alpha_{u', V_{k+1}}$, as well as the calculation of $B_{u, V_{k+1}}$, $B_{u', V_{k+1}}$ takes $O(n)$ time. The calculation of $\ell_{u, V_{k+1}}$, Ψ, and the subsequent ordering takes $O(1)$. (However, l is bounded by a constant independent of N; see [3, Remark 3.2]). Therefore, this case takes $O(n)$ time.

In total, the computation of $E_{k+1}^{(1)}$ takes $O(n) \operatorname{card} E_k = O(n^2)$ time, with the associated constants depending only on N. Moreover:

$$(5\text{–}1) \qquad \operatorname{card} E_{k+1}^{(1)} \leq \operatorname{card} E_k + 2 \sum_{e \in E_k} \operatorname{card} V_{k+1}(e)$$

Remark 5.1 Let $e, e' \in E_k$ satisfy the assumptions of the second case. Then by [3, Lemma 3.3], $V_{k+1}(e) \cap V_{k+1}(e') = \emptyset$.

5.2.3 Edges coming from points in $\mathcal{F}_k$

For each element u in $\mathcal{F}_k$ we calculate the closed ball $B_{u, V_{k+1}}$ and the associated line $\ell_{u, V_{k+1}}$ which takes $O(n)$ time. Calculate an isometric affine map $\Psi : \mathbb{R}^N \to \mathbb{R}^N$ that takes the line $\ell_{u, V_{k+1}}$ to the x_1–axis, and u to a point with zero first coordinate. This takes $O(1)$ of time. If there exists $u', u'' \in B_{u, V_{k+1}} \cap V_k$ such that $\langle \Psi(u') \rangle_1 < 0 < \langle \Psi(u'') \rangle_1$, then no new edges are obtained from u and we move to the next element of $\mathcal{F}_k$. Otherwise, we work as follows.

(1) If there exists no $u' \in B_{u,V_{k+1}} \cap V_k$ such that $\langle \Psi(u') \rangle_1 < 0$, then enumerate the points $u_1, \ldots, u_l \in \overline{B}(u, 2^{1-n_{k+1}} R_0)$ such that

$$\langle \Psi(u_1) \rangle_1 < \cdots < \langle \Psi(u_l) \rangle_1 < 0 = \langle \Psi(u) \rangle_1$$

and add the edges $\{u_1, u_2\}, \cdots, \{u_l, u\}$ in $E_{k+1}^{(2)}$.

(2) If there exists no $u' \in B_{u,V_{k+1}} \cap V_k$ such that $\langle \Psi(u') \rangle_1 > 0$, then enumerate the points $u_1, \ldots, u_l \in \overline{B}(u, 2^{1-n_{k+1}} R_0)$ such that

$$\langle \Psi(u_1) \rangle_1 > \cdots > \langle \Psi(u_l) \rangle_1 > 0 = \langle \Psi(u) \rangle_1$$

and add the edges $\{u_1, u_2\}, \cdots, \{u_l, u\}$ in $E_{k+1}^{(2)}$.

If none of the above is true, then no new edges are obtained from u and we move to the next element of $\mathcal{F}_k$. If both are true, we perform both of them. By [3, Remarks 3.4 and 3.5], we have that $l \leq 6$. Therefore, the computation of points u_i, their enumeration, and addition of edges takes $O(1)$ time. The set of all the points in $V_{k+1} \setminus V_k$ coming from both of these cases above (if any) is denoted by $V_{k+1}(u)$.

We repeat the same process for all points in $\mathcal{F}_k$ and the construction of $E_{k+1}^{(2)}$ is done in $(\text{card } \mathcal{F}_k) O(n) = O(n^2)$ time. Moreover:

$$(5\text{--}2) \qquad \text{card } E_{k+1}^{(2)} \leq 2 \sum_{u \in \mathcal{F}_k} \text{card } V_{k+1}(u)$$

Remark 5.2 By Badger, Naples, and Vellis [3, Lemma 3.6], if $u, u' \in \mathcal{F}_k$, then $V_{k+1}(u) \cap V_{k+1}(u') = \emptyset$. Moreover, if $e \in E_k$ and $u \in \mathcal{F}_k$, then $V_{k+1}(u) \cap V_{k+1}(e) = \emptyset$.

Before proceeding to the final set $E_{k+1}^{(3)}$ we enumerate:

$$\mathcal{N}_k = \{u_1, \ldots, u_l\}$$

5.2.4 Edges coming from $\mathcal{N}_k$: first step

Define the set:

$$V'_{k+1}(u_1) = B(u_1, C_0 2^{-n_{k+1}} R_0) \cap V_{k+1}$$

The construction of this set requires $O(n)$ time and its cardinality is $O(1)$. We now go over the set $E_{k+1}^{(1)} \cup E_{k+1}^{(2)}$ formed above and if a point $v \in V'_{k+1}(u_1)$ is contained in an edge of $E_{k+1}^{(1)} \cup E_{k+1}^{(2)}$, then we remove it from the set $V'_{k+1}(u_1)$ and in this way we form the set $V_{k+1}(u_1)$ in $O(n)$ time. By the doubling property of $\mathbb{R}^N$, card $V_{k+1}(u) = O(1)$.

Moreover, from the previous subsections we already have that every point in V_k is already contained in an edge of $E_{k+1}^{(1)} \cup E_{k+1}^{(2)}$. Therefore:

$$V_{k+1}(u_1) \subset V_{k+1} \setminus \left(V_k \cup \bigcup_{e \in E_k} V_{k+1}(e) \cup \bigcup_{u \in \mathcal{F}_k} V_{k+1}(u) \right)$$

If the set $V_{k+1}(u_1)$ is empty, then we are done with u_1 and we move to u_2. Otherwise, we construct the set:

$$\text{Flat}(k+1, u_1) = \bigcup_{u \in V_{k+1}(u_1)} \text{Flat}(u, V_k, V_{k+1})$$

Since there are $O(1)$ points in $V_{k+1}(u)$, by [Lemma 4.3](#) it takes a total of $O(n)$ time to determine $\bigcup_{u \in V_{k+1}(u_1)} \text{Flat}(u, V_k, V_{k+1})$. We make sure that the latter set does not contain the same edge twice and we denote it by $\text{Flat}(k+1, u_1)$. Since the cardinality of $\text{Flat}(k+1, u_1)$ is $O(1)$, this can be done in $O(1)$ time. Define $\tilde{V}_{k+1}(u_1)$ to be the union of $V'_{k+1}(u_1)$ and the set of all points in V_{k+1} that belong to an edge in $\text{Flat}(k+1, u_1)$.

Next, we form the graph $G_{k+1}(u_1) = (\tilde{V}_{k+1}(u_1), \text{Flat}(k+1, u_1))$. We apply the algorithm of [Lemma 2.1](#) which in $O(1)$ time returns the components of $G_{k+1}(u_1)$. If there is only one component, then we set $E_{k+1}(u_1) := \text{Flat}(k+1, u_1)$ (making sure in $O(1)$ time that no edge appears twice) and we add all edges of $E_{k+1}(u_1)$ (if any) in $E_{k+1}^{(3)}$ and we move to u_2. If there are p components with $p \geq 2$, then we fix a point $u_{1,i}$ in each component and we set

$$E_{k+1}(u_1) := \{\{u_{1,1}, u_{1,2}\}, \ldots, \{u_{1,1}, u_{1,p}\}\} \cup \text{Flat}(k+1, u_1)$$

and we add all edges of $E_{k+1}(u_1)$ in $E_{k+1}^{(3)}$.

Recall that each point in $V_{k+1}(u_1)$ can be in at most 2 edges of $\text{Flat}(k+1, u_1)$. Therefore, it is not hard to see that:

$$\text{card}(E_{k+1}(u_1)) \leq 2 \, \text{card}(V_{k+1}(u_1))$$

If $l = 1$, then the definition of $E_{k+1}^{(3)}$ is complete we move to [Section 5.2.6](#); otherwise and we move to u_2.

5.2.5 Edges coming from $\mathcal{N}_k$: inductive step

Suppose that we have already added edges to $E_{k+1}^{(3)}$ from vertices $u_1, \ldots, u_{p-1}$ with $p \leq l$. The definition of the set $V_{k+1}(u_p)$ is the same as with $V_{k+1}(u_1)$ with one

important exception: this time we go over the set $E_{k+1}^{(1)} \cup E_{k+1}^{(2)} \cup E_{k+1}^{(3)}$ instead of just $E_{k+1}^{(1)} \cup E_{k+1}^{(2)}$. Therefore,

$$(5\text{--}3) \qquad V_{k+1}(u_1) \subset V_{k+1} \setminus \left(V_k \cup \bigcup_{e \in E_k} V_{k+1}(e) \cup \bigcup_{u \in \mathcal{F}_k} V_{k+1}(u) \cup \bigcup_{i=1}^{p-1} V_{k+1}(u_i) \right).$$

The rest of the construction is verbatim the same as before and we add to $E_{k+1}^{(3)}$ a new (possibly empty) set of edges $E_{k+1}(u_p)$ with

$$(5\text{--}4) \qquad \operatorname{card}(E_{k+1}(u_p)) \le 2 \operatorname{card}(V_{k+1}(u_p)).$$

5.2.6 Definition of E_{k+1}

Going over the entire set of $\mathcal{N}_k$, we finally define $E_{k+1}^{(3)}$ in $O(n^2)$ time. From Remark 5.1, Remark 5.2, and (5–3), we have that the three sets $E_{k+1}^{(1)}$, $E_{k+1}^{(2)}$, $E_{k+1}^{(3)}$ are mutually disjoint and by estimates (5–1), (5–2) and (5–4):

$$\operatorname{card}(E_{k+1}^{(1)} \cup E_{k+1}^{(2)} \cup E_{k+1}^{(3)}) \le \operatorname{card} V_k + 2 \operatorname{card}(V_{k+1} \setminus V_k) \le 2 \operatorname{card} V_{k+1}$$

We now set $E_{k+1} = E_{k+1}^{(1)} \cup E_{k+1}^{(2)} \cup E_{k+1}^{(3)}$ (the enumeration takes $O(n)$ time) and the inductive step is complete.

5.3 Final Step:

To finish the proof of Theorem 1.1, we remark that for the final graph $G_m = (V_m, E_m)$ that we obtained, we have that $V_m = V$ and $\operatorname{card} E_m = O(n)$. Therefore, the algorithm of Lemma 2.2, produces the desired tour in $O(n^3)$ time. This completes the proof of Theorem 1.1.

5.4 Sharpness of exponent 3

Here we discuss why the time complexity $O(n^3)$ is sharp. We will do this by analyzing the computations of the flat pairs and flatness modules. Note that the complexity of $O(n^3)$ appears during many other steps of the algorithm as well (e.g. Lemma 2.1 and Lemma 2.2).

Let n be a positive integer and suppose that for all $i \in \{1, \dots, n\}$, $\operatorname{card} V_i = i$. At step k, we perform the calculation of sets $\operatorname{Flat}(v, V_k, V_{k+1})$ for each $v \in V_k$ which, by Remark 4.1, requires $C \operatorname{card} V_{k+1}$ time for each such v, and for some fixed constant $C > 0$. Therefore, the computation of all flat pairs at step k requires $C(\operatorname{card} V_k)(\operatorname{card} V_{k+1})$ computations. Therefore, our total computations over all steps are:

$$C \sum_{k=1}^{n-1} (\operatorname{card} V_k)(\operatorname{card} V_{k+1}) = C \sum_{k=1}^{n-1} k(k+1) \ge C \left(\frac{n}{2} \right)^2 \frac{n}{2} = C \frac{n^3}{8}$$

Since the calculation of the flatness modules for each net V_i is but a part of the algorithm, the total computation time is at least a fixed multiple of n^3.

Acknowledgements.

References

[1] **D L Applegate**, **R E Bixby**, **V Chvátal**, **W J Cook**, *The traveling salesman problem: A computational study*, Princeton Series in Applied Mathematics, Princeton University Press, Princeton, NJ (2006); https://doi.org/10.1515/9781400841103

[2] **S Arora**, *Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems*, J. ACM 45 (1998) 753–782; https://doi.org/10.1145/290179.290180

[3] **M Badger**, **L Naples**, **V Vellis**, *Hölder curves and parameterizations in the analyst's traveling salesman theorem*, Adv. Math. 349 (2019) 564–647; https://doi.org/10.1016/j.aim.2019.04.011

[4] **M Badger**, **R Schul**, *Multiscale Analysis of 1-rectifiable Measures II: Characterizations*, Anal. Geom. Metr. Spaces 5 (2017) 1–39; https://doi.org/10.1515/agms-2017-0001

[5] **Y Bartal**, **L-A Gottlieb**, **R Krauthgamer**, *The Traveling Salesman Problem: Low-Dimensionality Implies a Polynomial Time Approximation Scheme*, SIAM Journal on Computing 45 (2016) 1563–1581; https://doi.org/10.1137/130913328

[6] **R Bellman**, *Dynamic Programming Treatment of the Travelling Salesman Problem*, J. ACM 9 (1962) 61–63; https://doi.org/10.1145/321105.321111

[7] **N Christofides**, *Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem*, from: "Proceedings of Symposium on New Directions and Recent Results in Algorithms and Complexity" (1976); https://doi.org/10.1007/s43069-021-00101-z

[8] **M R Garey**, **R L Graham**, **D S Johnson**, *Some NP-Complete Geometric Problems*, from: "Proceedings of the Eighth Annual ACM Symposium on Theory of Computing", STOC '76, Association for Computing Machinery, New York, NY, USA (1976) 10–22; https://doi.org/10.1145/800113.803626

[9] **B Golden**, **L Bodin**, **T Doyle**, **W Stewart**, *Approximate Traveling Salesman Algorithms*, Operations Research 28 (1980) 694–711; https://doi.org/10.1287/opre.28.3.694

[10] **M Grigni**, **E Koutsoupias**, **C Papadimitriou**, *An approximation scheme for planar graph TSP*, from: "36th Annual Symposium on Foundations of Computer Science (Milwaukee, WI, 1995)", IEEE Comput. Soc. Press, Los Alamitos, CA (1995) 640–645; https://doi.org/10.1109/SFCS.1995.492665

[11] **X Gu**, **J H Lutz**, **E Mayordomo**, *Points on computable curves*, from: "2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)", IEEE (2006) 469–474; https://doi.org/10.1109/focs.2006.63

[12] **G Gutin**, **A P Punnen** (editors), *The traveling salesman problem and its variations*, volume 12 of *Combinatorial Optimization*, Kluwer Academic Publishers, Dordrecht (2002); https://doi.org/10.1007/0-306-48213-4_1

[13] **M Held**, **R M Karp**, *A dynamic programming approach to sequencing problems*, J. Soc. Indust. Appl. Math. 10 (1962) 196–210; https://doi.org/10.1137/0110015

[14] **D S Johnson**, **L A McGeoch**, *Experimental Analysis of Heuristics for the STSP*, from: "The Traveling Salesman Problem and Its Variations", (G Gutin, A P Punnen, editors), Springer US, Boston, MA (2007) 369–443; https://doi.org/10.1007/0-306-48213-4_9

[15] **P W Jones**, *Rectifiable sets and the traveling salesman problem*, Invent. Math. 102 (1990) 1–15; https://doi.org/10.1007/BF01233418

[16] **E L Lawler**, **J K Lenstra**, **A H G Rinnooy Kan**, **D B Shmoys** (editors), *The traveling salesman problem (A guided tour of combinatorial optimization)*, Wiley-Interscience Series in Discrete Mathematics, John Wiley & Sons, Ltd., Chichester (1985)

[17] **S Lin**, **B W Kernighan**, *An Effective Heuristic Algorithm for the Traveling-Salesman Problem*, Oper. Res. 21 (1973) 498–516; https://doi.org/10.1287/opre.21.2.498

[18] **J 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 (1999) 1298–1309; https://doi.org/10.1137/S0097539796309764

[19] **K Okikiolu**, *Characterization of subsets of rectifiable curves in $\mathbf{R}^n$*, J. London Math. Soc. (2) 46 (1992) 336–348; https://doi.org/10.1112/jlms/s2-46.2.336

[20] **C H Papadimitriou**, *The Euclidean travelling salesman problem is NP-complete*, Theoretical Computer Science 4 (1977) 237–244; https://doi.org/10.1016/0304-3975(77)90012-3

[21] **D J Rosenkrantz**, **R E Stearns**, **P M Lewis**, *Approximate algorithms for the traveling salesperson problem*, from: "15th Annual Symposium on Switching and Automata Theory (swat 1974)" (1974) 33–42; https://doi.org/10.1109/SWAT.1974.4

[22] **R Schul**, *Subsets of rectifiable curves in Hilbert space—the analyst's TSP*, J. Anal. Math. 103 (2007) 331–375; https://doi.org/10.1007/s11854-008-0011-y

Department of Mathematics, The University of Tennessee, Knoxville, TN 37966, USA

aramir13@utk.edu, vvellis@utk.edu

https://math.utk.edu/people/anthony-ramirez/,
https://sites.google.com/site/vyronvellis/home