Euclidean TSP in Narrow Strips

Discrete & Computational Geometry (2024) 71:1456–1506
https://doi.org/10.1007/s00454-023-00609-7

Euclidean TSP in Narrow Strips

Henk Alkema[1] · Mark de Berg[1] · Remco van der Hofstad[1] · Sándor Kisfaludi-Bak[2]

Received: 30 March 2022 / Revised: 20 June 2023 / Accepted: 29 September 2023 /
Published online: 8 January 2024

Abstract

We investigate how the complexity of EUCLIDEAN TSP for point sets P inside the strip $(-\infty, +\infty) \times [0, \delta]$ depends on the strip width δ. We obtain two main results.

- For the case where the points have distinct integer x-coordinates, we prove that a shortest bitonic tour (which can be computed in $O(n \log^2 n)$ time using an existing algorithm) is guaranteed to be a shortest tour overall when $\delta \leqslant 2\sqrt{2}$, a bound which is best possible.
- We present an algorithm that is fixed-parameter tractable with respect to δ. Our algorithm has running time $2^{O(\sqrt{\delta})}n + O(\delta^2 n^2)$ for sparse point sets, where each $1 \times \delta$ rectangle inside the strip contains $O(1)$ points. For random point sets, where the points are chosen uniformly at random from the rectangle $[0, n] \times [0, \delta]$, it has an expected running time of $2^{O(\sqrt{\delta})}n$. These results generalise to point sets P inside a hypercylinder of width δ. In this case, the factors $2^{O(\sqrt{\delta})}$ become $2^{O(\delta^{1-1/d})}$.

Editor in Charge: Kenneth Clarkson

The work in this paper is supported by the Netherlands Organisation for Scientific Research (NWO) through Gravitation-grant NETWORKS-024.002.003.
We would like to thank the anonymous DCG reviewer for their exceptionally thorough review and insightful comments.

Henk Alkema
h.y.alkema@tue.nl

Mark de Berg
m.t.d.berg@tue.nl

Remco van der Hofstad
r.w.v.d.hofstad@tue.nl

Sándor Kisfaludi-Bak
sandor.kisfaludi-bak@aalto.fi

1 Department of Mathematics and Computer Science, TU Eindhoven, Eindhoven, The Netherlands

2 Aalto University, Espoo, Finland

Keywords Computational geometry · Euclidean TSP · Bitonic TSP · Fixed-parameter tractable algorithms

Mathematics Subject Classification 68Q25 · 68W40

1 Introduction

In the TRAVELING SALESMAN PROBLEM one is given an edge-weighted complete graph and the goal is to compute a tour—a simple cycle visiting all nodes—of minimum total weight. Due to its practical as well as theoretical importance, the TRAVELING SALESMAN PROBLEM and its many variants are among the most famous problems in computer science and combinatorial optimization. In this paper we study the Euclidean version of the problem. In EUCLIDEAN TSP the input is a set P of n points in $\mathbb{R}^d$, and the goal is to compute a minimum-length tour visiting each point. EUCLIDEAN TSP in the plane was proven to be NP-hard in the 1970s [18, 23]. Around the same time, Christofides [4] gave an elegant (3/2)-approximation algorithm, which works in any metric space. For a long time it was unknown if EUCLIDEAN TSP is APX-hard, until Arora [2], and independently Mitchell [22], presented a PTAS. Mitchell's algorithm works for the planar case, while Arora's algorithm also works in higher dimensions. Rao and Smith [24] later improved the running time of Arora's PTAS, obtaining a running time of $2^{(1/\varepsilon)^{O(d)}}n + (1/\varepsilon)^{O(d)}n \log n$ in $\mathbb{R}^d$.

We are interested in exact algorithms for EUCLIDEAN TSP. As mentioned, the problem is already NP-hard in the plane. Unlike the general (metric) version, however, it can be solved in *subexponential* time, that is, in time $2^{o(n)}$. In particular, Kann [21] and Hwang et al. [19] presented algorithms with $n^{O(\sqrt{n})}$ running time. Smith and Wormald [29] gave a subexponential algorithm that works in any (fixed) dimension; its running time in $\mathbb{R}^d$ is $n^{O(n^{1-1/d})}$. Very recently De Berg et al. [12] improved this to $2^{O(n^{1-1/d})}$, which is tight up to constant factors in the exponent, under the Exponential-Time Hypothesis (ETH) [20].

There has also been considerable research on special cases of EUCLIDEAN TSP that are polynomial-time solvable. One example is BITONIC TSP, where the goal is to find a shortest *bitonic* tour. (A tour is bitonic if any vertical line crosses it at most twice; here the points from the input set P are assumed to have distinct x-coordinates.) It is a classic exercise [5] to prove that BITONIC TSP can be solved in $O(n^2)$ time by dynamic programming. De Berg et al. [10] showed how to speed this algorithm up to $O(n \log^2 n)$. When P is in convex position, then the convex hull of P is a shortest tour and so one can solve EUCLIDEAN TSP in $O(n \log n)$ time [9]. Deineko et al. [14] studied the case where the points need not all be on the convex hull; the points inside the convex hull, however, are required to be collinear. Their algorithm runs in $O(n^2)$ time. Deineko and Woeginger [13] extended this to the case where the points in the interior of the convex hull lie on k parallel lines, obtaining an $O(n^{k+2})$ algorithm. These results generalize earlier work by Cutler [6] and Rote [27] who consider point sets lying on three, respectively k, parallel lines. Deineko et al. [15] gave a fixed-parameter tractable algorithm for EUCLIDEAN TSP where the parameter

k is the number of points inside the convex hull, with running time $O(2^k k^2 n)$. Finally, Reinhold [25] and Sanders [28] proved that when there exists a collection of disks centered at the points in P whose intersection graph is a single cycle—this is called the necklace condition—then the tour following the cycle is optimal. Edelsbrunner et al. [16] gave an $O(n^2 \log n)$ algorithm to verify whether such a collection of disks exists (and, if so, find one).

Our Contribution. The computational complexity of EUCLIDEAN TSP in $\mathbb{R}^d$ is $2^{\Theta(n^{1-1/d})}$ (for $d \geqslant 2$), assuming ETH. Thus the complexity depends heavily on the dimension d. This is most pronounced when we compare the complexity for $d = 2$ with the trivial case $d = 1$: in the plane EUCLIDEAN TSP takes $2^{\Theta(\sqrt{n})}$ time in the worst case, while the 1-dimensional case is trivially solved in $O(n \log n)$ time by sorting the points. We study the complexity of EUCLIDEAN TSP for planar point sets that are "almost 1-dimensional". In particular, we assume that the point set P is contained in the hypercylinder $\mathbb{R} \times \text{Ball}^{d-1}(\delta/2)$, where $\text{Ball}^{d-1}(\delta/2)$ denotes the closed $(d-1)$-dimensional ball with radius $\delta/2$, for some relatively small δ and some arbitrary but fixed $d \geqslant 2$. We investigate how the complexity of EUCLIDEAN TSP depends on the parameter δ. As any instance of EUCLIDEAN TSP can be scaled to fit inside a hypercylinder, we need to make some additional restriction on the input. We consider three scenarios.

- *Integer x-coordinates, with $d = 2$.* BITONIC TSP can be solved in $O(n \log^2 n)$ time [10]. It is natural to conjecture that for points with distinct integer x-coordinates inside a sufficiently narrow strip, an optimal bitonic tour is a shortest tour overall. We give a (partially computer-assisted) proof that this is indeed the case: we prove that when $\delta \leqslant 2\sqrt{2}$ an optimal bitonic tour is optimal overall, and we show that the bound $2\sqrt{2}$ is best possible.
- *Sparse point sets.* We generalize the case of integer x-coordinates to the case where the drum $[x, x+1] \times \text{Ball}^{d-1}(\delta/2)$ contains $O(1)$ points for all $x \in \mathbb{R}$. Furthermore, we investigate how the complexity of EUCLIDEAN TSP grows with δ. We show that for sparse point sets in $\mathbb{R}^2$ an optimal tour must be k-tonic—a tour is k-tonic if it intersects any vertical line at most k-times—for $k = O(\sqrt{\delta})$. This suggests that one might be able to use a dynamic-programming algorithm similar to the ones for points on k parallel lines [13, 27]. The latter algorithms run in $O(n^k)$ time, suggesting that a running time of $n^{O(\sqrt{\delta})}$ is achievable for $d = 2$ in our case. We give a much more efficient algorithm, which is fixed-parameter tractable (and subexponential) with respect to the parameter δ, and which generalizes to $d > 2$. Its running time for sparse point sets in $\mathbb{R}^d$ is $2^{O(\delta^{1-1/d})}n + O(\delta^2 n^2)$.
- *Random point sets.* In the third scenario the points in P are drawn independently and uniformly at random from the hypercylinder $[0, n] \times \text{Ball}^{d-1}(\delta/2)$. For this case we prove that a very similar algorithm to the algorithm for sparse point sets has an expected running time of $2^{O(\delta^{1-1/d})}n$, which is linear if $\delta = O(1)$.

Notation and Terminology. Let $P := \{p_1, \ldots, p_n\}$ be a set of points in a d-dimensional hypercylinder with radius $\delta/2$ —we call such a hypercylinder a *δ-cylinder*—which we assume without loss of generality to be $\mathbb{R} \times \text{Ball}^{d-1}(\delta/2)$. We denote the x-coordinate of a point $p \in \mathbb{R}^d$ by $x(p)$, and its other coordinates by

$y_1(p), ..., y_{d-1}(p)$. To simplify the notation, we also write x_i for $x(p_i)$. We sort the points in P such that $x_i \leqslant x_{i+1}$ for all $1 \leqslant i < n$. We define the *spacing* $\Delta_i := x_{i+1} - x_i$.

For two points $p, q \in \mathbb{R}^d$, we write pq to denote the *directed* edge from p to q. Paths are written as lists of points, so $(q_1, q_2, \ldots, q_m)$ denotes the path consisting of the edges $q_1q_2, \ldots, q_{m-1}q_m$. All points in a path must be distinct, except possibly $q_1 = q_m$ in which case the path is a tour. The length of an edge pq is denoted by $|pq|$, and the total length of a set E of edges is denoted by $\|E\|$. Finally, a *separator* is a hyperplane orthogonal to the x-axis, not containing any of the points in P.

2 Bitonicity for Points with Integer *x*-Coordinates

In this section we consider the case where $d = 2$ and the points in P have distinct integer x-coordinates, which implies that $\Delta_i \in \mathbb{N}^+$ for all i. Note that separators in $\mathbb{R}^2$ are vertical lines. For simplicity, we will write $y(p) := y_1(p)$, and $y_i := y(p_i)$. We will also assume the points are in the δ-*strip* $\mathbb{R} \times [0, \delta]$.

For our purposes, two separators s, s' that induce the same partitioning of P are equivalent. Therefore, we can define $\mathcal{S} := \{s_1, \ldots, s_{n-1}\}$ as the set of all combinatorially distinct separators, obtained by taking one separator s_i between any two points p_i, p_{i+1}. Let E be a set of edges with endpoints in P. The *tonicity of* E *at a separator* s, written as $\text{ton}(E, s)$, is the number of edges in E crossing s. We say that a set E has *lower tonicity* than a set F of edges, denoted by $E \preccurlyeq F$, if $\text{ton}(E, s_i) \leqslant \text{ton}(F, s_i)$ for all $s_i \in \mathcal{S}$. The set E has *strictly lower tonicity*, denoted by $E \prec F$, if there also exists at least one i for which $\text{ton}(E, s_i) < \text{ton}(F, s_i)$. Finally, we call a set E of edges k-*tonic* if $\text{ton}(E, s_i) \leqslant k$ for all $s_i \in \mathcal{S}$. We will use the terms *monotonic* and *bitonic* to denote 1-tonic and 2-tonic, respectively. We say E has *exact tonicity* k if E is k-tonic, but not $(k-1)$-tonic.

The goal of this section is to prove the following theorem.

Theorem 2.1 *Let P be a set of points with distinct and integer x-coordinates in a δ-strip. When $\delta \leqslant 2\sqrt{2}$, a shortest bitonic tour on P is a shortest tour overall. Moreover, for any $\delta > 2\sqrt{2}$ there is a point set P in a δ-strip such that a shortest bitonic tour on P is not a shortest tour overall.*

The construction for the case $\delta > 2\sqrt{2}$ is shown in Fig. 1. It is easily verified that, up to symmetrical solutions, the tours T_1 and T_2 are the only candidates for the shortest tour. Observe that $\|T_2\| - \|T_1\| = |p_1p_4| - |p_4p_5| = 3 - \sqrt{1+\delta^2}$. Hence, for $\delta > 2\sqrt{2}$ we have $\|T_2\| < \|T_1\|$, which proves the lower bound of Theorem 2.1. The remainder of the section is devoted to proving the first statement.

Let P be a point set in a δ-strip for $\delta = 2\sqrt{2}$, where all points in P have distinct integer x-coordinates. Among all shortest tours on P, let T_{opt} be one that is minimal with respect to the $\preccurlyeq$-relation; T_{opt} exists, since the number of different tours on P is finite. We claim that T_{opt} is bitonic, proving the upper bound of Theorem 2.1.

Suppose for a contradiction that T_{opt} is not bitonic. Let $s^* \in \mathcal{S}$ be the rightmost separator for which $\text{ton}(T_{\text{opt}}, s^*) > 2$. Note that for any two consecutive separators s_i, s_{i+1}, the difference in tonicity at those separators is at most 2. Hence, $\text{ton}(T_{\text{opt}}, s^*) = 4$.

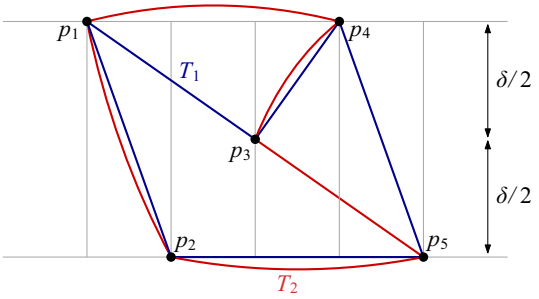


Fig. 1 Construction for $\delta > 2\sqrt{2}$ for Theorem 2.1. The grey vertical segments are at distance 1 from each other. If $\delta > 2\sqrt{2}$ then T_1, the shortest bitonic tour (in blue), is longer than T_2, the shortest non-bitonic tour (in red)

Let F be the four edges of T_{opt} crossing s^*, and let E be the remaining set of edges of T_{opt}. Let Q be the set of endpoints of the edges in F. We will argue that there exists a set F' of edges with endpoints in Q such that $E \cup F'$ is a tour and (i) $\|F'\| < \|F\|$, or (ii) $\|F'\| = \|F\|$ and $F' \prec F$. We will call such an F' *superior to* F. Option (i) contradicts that T_{opt} is a shortest tour. Since $E \cup F' \prec E \cup F$ if and only if $F' \prec F$, (ii) contradicts that T_{opt} is a shortest tour that is minimal with respect to $\preccurlyeq$. Hence, proving that such a superior set F' exists finishes the proof.

The remainder of the proof proceeds in two steps. In the first step we move the points in Q, obtaining a set $\overline{Q}$ with consecutive integer coordinates and an edge set $\overline{F}$. These will be such that if an $\overline{F}'$ superior to $\overline{F}$ exists, then there also exists an F' superior to F. In the second step we then give a computer-assisted proof that the desired set $\overline{F}'$ exists.

Step 1: Finding a suitable $\overline{Q}$ with consecutive x-coordinates. Let T_{opt}, s^*, E, F and Q be defined as above. We assume without loss of generality that the x-coordinate of s^* is equal to $x^* + \frac{1}{2}$, where x^* is the largest integer such that the line $x = x^* + \frac{1}{2}$ intersects all four edges in F. Since the actual edges in E are not important for our arguments, we replace them by abstract "connections" specifying which pairs of endpoints of the edges in F are connected by paths of edges in E. It will be convenient to duplicate the points in Q that are shared endpoints of two edges in F, and add a connection between the two copies; see Fig. 2. We denote the set of connections obtained in this way by $\widetilde{E}$, and we call $\widetilde{E}$ the *connectivity pattern* of F (in $E \cup F$).

Next we show how to move the points in Q such that the modified set $\overline{Q}$ uses consecutive x-coordinates. Recall that $s^* : x = x^* + \frac{1}{2}$ is a separator that intersects all edges in F. Let Q_{left} and Q_{right} be the subsets of points from Q lying to the left and right of s^*, respectively. We will move the points in Q_{left} such that they will get consecutive x-coordinates with the largest one being equal to x^*, while the points in Q_{right} will get consecutive x-coordinates with the smallest one being $x^* + 1$.

We move the points in Q_{left} as follows. Let $z \leqslant x^*$ be the largest x-coordinate currently not in use by any of the points in Q_{left}. If Q_{left} lies completely to the right of the line $\ell(z) : x = z$, then we are done: the set of x-coordinates used by points in Q_{left} is $\{z+1, \ldots, x^*\}$. Otherwise, we take any point to the left of $\ell(z)$, and we move

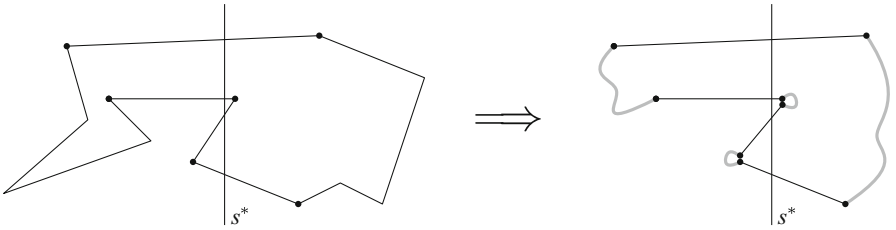


Fig. 2 Replacing the paths connecting endpoints of edges in F by abstract connections. The copies of duplicated shared endpoints are slightly displaced in the figure to be able to distinguish them, but they are actually coinciding

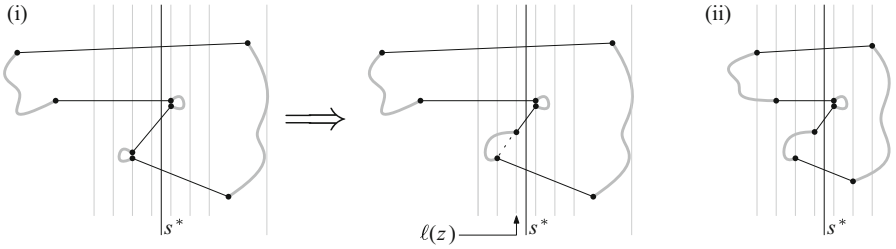


Fig. 3 The process of moving the points in Q. Grey vertical lines have integer x-coordinates. **i** Moving a point in Q_{left} so that it gets x-coordinate z. **ii** A possible configuration after Q_{left} and Q_{right} have been treated

it along its edge $e \in F$ to the point $e \cap \ell(z)$; see Fig. 3i. This process is repeated until the points of Q_{left} have consecutive x-coordinates. Note that duplicate x-coordinates caused by duplicating shared endpoints can still exist, if neither of the duplicates was moved.

After moving the points in Q_{left} we treat Q_{right} in a similar manner; the only difference is that now we define $z > x^*$ to be the smallest x-coordinate currently not in use by any of the points in Q_{right}. Figure 3ii shows the final result for the example in part (i) of the figure.

Before we prove that this procedure preserves the desired properties, two remarks are in order about the process described above. First, in each iteration we may have different choices for the edge e crossing $\ell(z)$, and the final result depends on these choices. Second, when we move a point in Q to a new location, then the new x-coordinate is not used by Q but it may already be used by points in $P \setminus Q$. Neither of these facts causes any problems for the coming arguments.

Let $\overline{Q}$ be the set of points from Q after they have been moved to their new locations, and let $\overline{F}$ be the set of edges from F after the move. With a slight abuse of notation we still use $\widetilde{E}$ to specify the connectivity pattern on $\overline{F}$, which is simply carried over from F. The following lemma shows that we can use $\overline{F}$, $\overline{Q}$ and $\widetilde{E}$ in Step 2 of the proof.

Lemma 2.2 *Let $E, F, Q, \widetilde{E}, \overline{F}, \overline{Q}$ be defined as above. Let $\overline{F}'$ be any set of edges (with endpoints in $\overline{Q}$) superior to $\overline{F}$, such that $\widetilde{E} \cup \overline{F}'$ is a tour. Then there is a set of edges F' (with endpoints in Q) superior to F such that $E \cup F'$ is a tour.*

Proof We define F' in the obvious way, by simply taking $\overline{F}'$ and replacing each endpoint (which is a point in $\overline{Q}$) by the corresponding point in Q. Clearly $E \cup F'$ forms a tour if $\widetilde{E} \cup \overline{F}'$ forms a tour.

Suppose $\overline{F}'$ is superior to $\overline{F}$. We will show that F' is superior to F. We will do this by first proving that $\|F\| - \|F'\| \geqslant \|\overline{F}\| - \|\overline{F}'\|$. Then, we prove that $F' \preccurlyeq F$. Finally, we prove that if there exists a separator $\overline{s} \in \mathcal{S}$ such that $\mathrm{ton}(\overline{F}', \overline{s}) < \mathrm{ton}(\overline{F}, \overline{s})$, then $\mathrm{ton}(F', s^*) < \mathrm{ton}(F, s^*)$.

Recall that each edge $\overline{e} \in \overline{F}$ is obtained from the corresponding edge $e \in F$ by moving one or both endpoints along the edge e itself. Also recall that we duplicated shared endpoints of edges in F, so if we move a point in Q, then we move the endpoint of a single edge in F. Hence,

$$\|F\| - \|\overline{F}\| = \text{total distance over which the points in } Q \text{ are moved.}$$

Since we added a connection to $\widetilde{E}$ between the two copies of a shared endpoint, each point in Q is incident to exactly one connection in $\widetilde{E}$ and, hence, to exactly one edge in F'. This means that if we move a point in Q, then we move the endpoint of a single edge in F', so

$$\|F'\| - \|\overline{F}'\| \leqslant \text{total distance over which the points in } Q \text{ are moved.}$$

We conclude that $\|F\| - \|F'\| \geqslant \|\overline{F}\| - \|\overline{F}'\|$.

Next, we claim that $F' \preccurlyeq F$. We will prove this by showing that F is maximal with respect to $\preccurlyeq$. Let s be any separator. Let G be any edge set on Q such that $\widetilde{E} \cup G$ forms a tour. Now, the tonicity of G at s is bounded by the minimum of the number of points of Q to the left of s and the number of points of Q to the right of s. Note that F attains this bound for every separator s. We conclude that F is maximal with respect to $\preccurlyeq$, and therefore, $F' \preccurlyeq F$ indeed holds.

Finally, let $\overline{s} \in \mathcal{S}$ be such that $\mathrm{ton}(\overline{F}', \overline{s}) < \mathrm{ton}(\overline{F}, \overline{s})$. We will show that $\mathrm{ton}(F', s^*) < \mathrm{ton}(F, s^*)$. Without loss of generality, let $\overline{s}$ either be to the left of s^*, or s^* itself. Since $\mathrm{ton}(\overline{F}', \overline{s}) < \mathrm{ton}(\overline{F}, \overline{s})$, the edge set $\overline{F}'$ contains an edge e fully to the left of $\overline{s}$: otherwise, the tonicity of $\overline{F}'$ at $\overline{s}$ would be equal to the number of points to the left of $\overline{s}$, which is equal to the tonicity of $\overline{F}$ at $\overline{s}$. Furthermore, since

$$\mathrm{ton}(F, s^*) = \mathrm{ton}(\overline{F}, s^*)$$

and

$$\mathrm{ton}(F', s^*) = \mathrm{ton}(\overline{F}', s^*),$$

we conclude that $\mathrm{ton}(F', s^*) < \mathrm{ton}(F, s^*)$.

Recall that by the definition of superior, F' is superior to F if and only if (a) $\|F'\| < \|F\|$, or (b) $\|F'\| = \|F\|$ and $F' \prec F$. We have proven that (i) $\|F\| - \|F'\| \geqslant \|\overline{F}\| - \|\overline{F}'\|$, (ii) $F' \preccurlyeq F$, and (iii) if $\mathrm{ton}(\overline{F}', s) < \mathrm{ton}(\overline{F}, s)$ for some separator $s \in \mathcal{S}$, then $\mathrm{ton}(F', s^*) < \mathrm{ton}(F, s^*)$.

Suppose that $\overline{F}'$ is superior to $\overline{F}$. If $\|\overline{F}'\| < \|\overline{F}\|$, then by (i), we get that (a) holds. Else, $\|\overline{F}'\| = \|\overline{F}\|$, so by (i), we get that either (a) holds, or $\|F'\| = \|F\|$, satisfying the first requirement of (b). Furthermore, since in this case $\overline{F}' \prec \overline{F}$, there exists a separator $\overline{s}$ satisfying $\text{ton}(\overline{F}', \overline{s}) < \text{ton}(\overline{F}, \overline{s})$. By (iii), we have $\text{ton}(F', s^*) < \text{ton}(F, s^*)$. Combining this with (ii) gives us that $F' \prec F$, which is the second requirement of (b).

We conclude that if $\overline{F}'$ is superior to $\overline{F}$, then F' is superior to F. □

Step 2: Finding the set F'. The goal of Step 2 of the proof is the following: given the tour $T_{\text{opt}} = E \cup F$ inside a δ-strip of width $\delta = 2\sqrt{2}$, show that there exists a set F' of edges such that $E \cup F'$ is a tour and F' is superior to F. Lemma 2.2 implies that we may work with $\widetilde{E}$ and $\overline{F}$ instead of E and F (and then find $\overline{F}'$ instead of F').

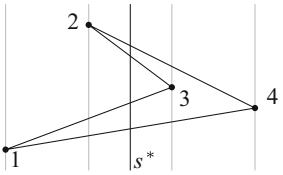
In Step 1 we duplicated shared endpoints of edges in E. We now merge these two copies again if they are still at the same location. This will always be the case for the shared endpoint immediately to the right of the separator s^*: we picked $s^* : x = x^* + \frac{1}{2}$ such that there is a shared endpoint at $x = x^* + 1$ (since s^* is the rightmost separator with $\text{ton}(T_{\text{opt}}, s^*) = 4$), and the copies of this endpoint will not be moved. So if n_{left} and n_{right} denote the number of distinct endpoints to the right and left of s^*, respectively, then $n_{\text{right}} \in \{2, 3\}$ and $n_{\text{left}} \in \{2, 3, 4\}$. We thus have six cases in total for the pair $(n_{\text{left}}, n_{\text{right}})$, as depicted in Fig. 4. Each of the six cases has several subcases, depending on the left-to-right order of the vertices inside the gray rectangles in the figure. For any fixed ordering we can still vary the y-coordinates in the range $[0, \delta]$. This may lead to scenarios where different sets $\overline{F}'$ are required. We handle this potentially huge amount of cases in a computer-assisted manner, using an automated prover *FindShorterTour*$(n_{\text{left}}, n_{\text{right}}, \overline{F}, \widetilde{E}, X, \delta, \varepsilon)$. The input parameter X is an array where $X[i]$ specifies the set from which the x-coordinate of the i-th point in the given scenario may be chosen. Here, we assume w.l.o.g. that $x(s^*) = -1/2$; see Fig. 4. The smaller the precision parameter ε is, the more precise the output will be. Its precise role will be explained below.

The output of *FindShorterTour* is a list of *scenarios* and an *outcome* for each scenario. A scenario contains for each point q an x-coordinate $x(q)$ from the set of allowed x-coordinates for q, and a range y-range$(q) \subseteq [0, 2\sqrt{2}]$ for its y-coordinate. This y-range is an interval of length at least $\varepsilon/2$. The outcome of a scenario is either SUCCESS or FAIL. An outcome SUCCESS means that a set $\overline{F}'$ has been found with the desired properties: $\widetilde{E} \cup \overline{F}'$ is a tour, and for all possible instantiations of the scenario—that is, all choices of y-coordinates from the y-ranges in the scenario—we have $\|\overline{F}'\| < \|\overline{F}\|$. An outcome FAIL means that such an $\overline{F}'$ has not been found. It does not guarantee that such an $\overline{F}'$ does not exist for this scenario.

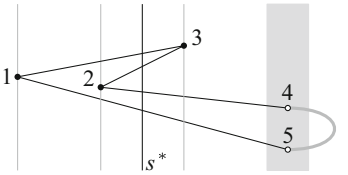
The list of scenarios is complete in the sense that for any instantiation of the input case there is a scenario that covers it.

FindShorterTour works brute-force. It checks all possible combinations of x-coordinates and subdivides the y-coordinate ranges, until a suitable $\overline{F}'$ can be found or until the y-ranges have length smaller than the precision parameter ε. The implementation details of the procedure are in Appendix A.

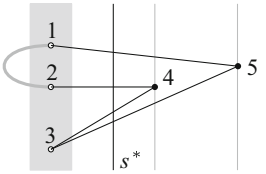
Note that case $(n_{\text{left}}, n_{\text{right}}) = (2, 3)$ in Fig. 4 is a subcase of case $(n_{\text{left}}, n_{\text{right}}) = (3, 2)$, if we exchange the roles of the points lying to the left and to the right of s^*,



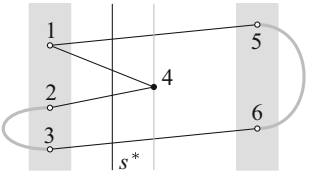
- $(n_{\text{left}}, n_{\text{right}}) = (2, 2)$
- $\overline{F} = \{(1, 3), (1, 4), (2, 3), (2, 4)\}$
- $\widetilde{E} = \emptyset$
- $X = [\{-2\}, \{-1\}, \{0\}, \{1\}]$



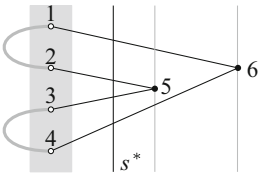
- $(n_{\text{left}}, n_{\text{right}}) = (2, 3)$
- $\overline{F} = \{(1, 3), (1, 5), (2, 3), (2, 4)\}$
- $\widetilde{E} = \{(4, 5)\}$
- $X = [\{-2\}, \{-1\}, \{0\}, \{1, 2\}, \{1, 2\}]$



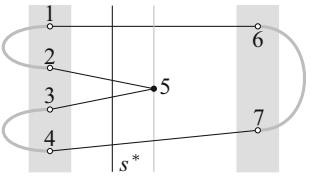
- $(n_{\text{left}}, n_{\text{right}}) = (3, 2)$
- $\overline{F} = \{(1, 4), (1, 5), (2, 4), (3, 5)\}$
- $\widetilde{E} = \{(2, 3)\}$
- $X = [\{-3, -2, -1\}, \{-3, -2, -1\}, \{-3, -2, -1\}, \{0\}, \{1\}]$



- $(n_{\text{left}}, n_{\text{right}}) = (3, 3)$
- $\overline{F} = \{(1, 4), (1, 5), (2, 4), (3, 6)\}$
- $\widetilde{E} = \{(2, 3), (5, 6)\}$
- $X = [\{-3, -2, -1\}, \{-3, -2, -1\}, \{-3, -2, -1\}, \{0\}, \{1, 2\}, \{1, 2\}]$

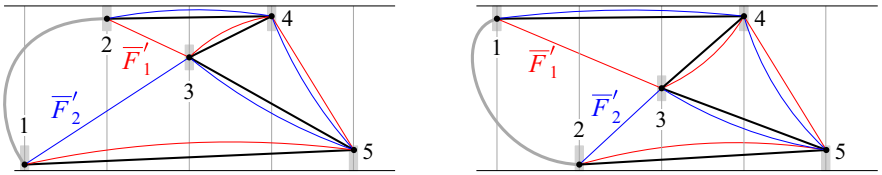


- $(n_{\text{left}}, n_{\text{right}}) = (4, 2)$
- $\overline{F} = \{(1, 6), (2, 5), (3.5), (4, 6)\}$
- $\widetilde{E} = \{(1, 2), (3, 4)\}$
- $X = [\{-4, ..., -1\}, \{-4, ..., -1\}, \{-4, ..., -1\}, \{-4, ..., -1\}, \{0\}, \{1\}]$



- $(n_{\text{left}}, n_{\text{right}}) = (4, 3)$
- $\overline{F} = \{(1, 6), (2, 5), (3.5), (4, 7)\}$
- $\widetilde{E} = \{(1, 2), (3, 4), (6, 7)\}$
- $X = [\{-4, ..., -1\}, \{-4, ..., -1\}, \{-4, ..., -1\}, \{-4, ..., -1\}, \{0\}, \{1, 2\}, \{1, 2\}]$

Fig. 4 The six different cases that result after applying Step 1 of the proof. Points indicated by filled disks have a fixed x-coordinate. The left-to-right order of points drawn inside a grey rectangle, on the other hand, is not known yet. The vertical order of the edges is also not fixed, as the points can have any y-coordinate in the range $[0, 2\sqrt{2}]$



- $X = [\{-3\}, \{-2\}, \{-1\}, \{0\}, \{1\}]$
- $Y = [[0, 0.01], [2.82, 2\sqrt{2}], [1.61, 1.62], [2.82, 2\sqrt{2}], [0, 0.01]]$

- $X = [\{-3\}, \{-2\}, \{-1\}, \{0\}, \{1\}]$
- $Y = [[2.82, 2\sqrt{2}], [0, 0.01], [1.41, 1.42], [2.82, 2\sqrt{2}], [0, 0.01]]$

Fig. 5 Two scenarios covering all subscenarios where the automated prover fails, up to symmetries. Each point has a fixed x-coordinate and a y-range specified by the array Y; the resulting possible locations are shown as small grey rectangles (drawn larger than they actually are for visibility). For all subscenarios, at least one of $\overline{F}'_1$ (in red) and $\overline{F}'_2$ (in blue) is at most as long as $\overline{F}$ (in black)

swap the labels of points 5 and 1, and swap the labels of points 2 and 4. Hence, we ignore this subcase and run our automated prover on the remaining five cases, where we set $\varepsilon := 0.001$. It successfully proves the existence of a suitable set $\overline{F}'$ in four cases; the case where the prover fails is the case $(n_{\text{left}}, n_{\text{right}}) = (3, 2)$. For this case, it fails for the two scenarios depicted in Fig. 5. All other scenarios for these cases are handled successfully (up to symmetries). For both scenarios in Fig. 5 we consider two alternatives for the set $\overline{F}'$: the set $\overline{F}'_1$ shown in red in Fig. 5, and the set $\overline{F}'_2$ shown in blue. We will show that in any instantiation of both scenarios, either $\overline{F}'_1$ or $\overline{F}'_2$ is at least as short as $\overline{F}$. Since both alternatives are bitonic, this finishes the proof.

For $1 \leqslant i \leqslant 5$, let q_i be the point labeled i in the left scenario in Fig. 5. We first argue that we can assume without loss of generality that $y(q_2) = y(q_4) = 2\sqrt{2}$ and $y(q_1) = y(q_5) = 0$. To this end, consider an arbitrary instantiation of this scenario, and imagine moving q_2 and q_4 up to the line $y = 2\sqrt{2}$, and moving q_1 and q_5 down to the line $y = 0$. It suffices to show, for $i \in \{1, 2\}$, that if we have $\|\overline{F}'_i\| \leqslant \|\overline{F}\|$ after the move, then we also have $\|\overline{F}_i\| \leqslant \|\overline{F}\|$ before the move. To see why this indeed holds, we will need the following observation.

Observation 2.3 *Let p be a point in $\mathbb{R}^2$. Suppose we move p in direction $\vec{v}$ at unit speed. Let $p(t)$ denote the position of point p at time t. Let $\vec{p}(t)$ be the corresponding vector. Then $\frac{d}{dt}|\vec{p}(t)| = \frac{\vec{p}(t)\cdot v}{|\vec{p}(t)|}$, where '$\cdot$' denotes the inner product. This also equals the cosine of the angle between $\vec{p}(t)$ and $\vec{v}$.*

Proof Assume without loss of generality that $\vec{v} = (1, 0)$. Then

$$\begin{aligned}\frac{d}{dt}|\vec{p}(t)| &= \frac{d}{dt}\sqrt{x(p(t))^2 + y(p(t))^2} \\ &= \frac{d}{dt}\sqrt{(x(p) + t)^2 + y(p)^2} \\ &= \frac{x(p) + t}{\sqrt{(x(p) + t)^2 + y(p)^2}}\end{aligned}$$

$$= \frac{\vec{p}(t) \cdot \vec{v}}{|\vec{p}(t)|}.$$

Finally, for any vector $\vec{v}_1, \vec{v}_2$ we have $\vec{v}_1 \cdot \vec{v}_2 = |\vec{v}_1||\vec{v}_2| \cos\alpha$, where α is the angle between $\vec{v}_1$ and $\vec{v}_2$. Since $\vec{v}$ is a unit-length vector, this finalizes our argument. □

This leads us to the following corollary.

Corollary 2.4 *Let a, b, c be three points. Let ℓ be the vertical line through c, and suppose we move c downwards along ℓ. Let α be the smaller angle between ac and ℓ if $y(c) < y(a)$, and the larger angle otherwise. Let β be the smaller angle between bc and ℓ if $y(c) < y(b)$, and the larger angle otherwise. Suppose $\alpha < \beta$ throughout the move. Then the move increases $|ac|$ more than it increases $|bc|$.*

Proof From Observation 2.3 we can directly conclude that at any time during the move, the derivative of $|ac|$ is strictly larger than the derivative of $|bc|$. □

We can now repeatedly apply Corollary 2.4 to our current scenario, the left scenario in Fig. 5. Let us first compare $\|\overline{F}'_1\|$ to $\|\overline{F}\|$. Moving q_1 downwards to $y = 0$ changes both by the same amount. We can apply Corollary 2.4 to q_2: by moving it upwards, $|q_2q_3|$ increases more than $|q_2q_4|$, thus $\|\overline{F}'_1\| - \|\overline{F}\|$ increases. Then, moving q_4 upwards, $|q_2q_4|$ decreases while $|q_4q_5|$ increases, and therefore $\|\overline{F}'_1\| - \|\overline{F}\|$ increases once more. Finally, we can apply Corollary 2.4 to show that if we move q_5 downwards, $|q_4q_5|$ increases more than $|q_3q_5|$, and therefore increasing $\|\overline{F}'_1\| - \|\overline{F}\|$ again. Since none of the moves decreased $\|\overline{F}'_1\| - \|\overline{F}\|$, we conclude that if $\|\overline{F}'_1\| \leqslant \|\overline{F}\|$ after the move, then we also have $\|\overline{F}_1\| \leqslant \|\overline{F}\|$ before the move. We can prove analogously that if $\|\overline{F}'_2\| \leqslant \|\overline{F}\|$ after the move, then we also have $\|\overline{F}_2\| \leqslant \|\overline{F}\|$ before the move.

So now assume $y(q_2) = y(q_4) = 2\sqrt{2}$ and $y(q_1) = y(q_5) = 0$. Let $y := y(q_3)$. If $y \geqslant (8\sqrt{2})/7$ then

$$|q_2q_3| + |q_4q_5| = \sqrt{1 + (2\sqrt{2} - y)^2} + 3 \leqslant 2 + \sqrt{4 + y^2} = |q_2q_4| + |q_3q_5|,$$

so $\|\overline{F}'_1\| \leqslant \|\overline{F}\|$. On the other hand, if $y \leqslant (8\sqrt{2})/7$ then

$$|q_1q_3| + |q_4q_5| = \sqrt{4 + y^2} + 3 \leqslant \sqrt{1 + (2\sqrt{2} - y)^2} + 4 = |q_3q_4| + |q_1q_5|,$$

so $\|\overline{F}'_2\| \leqslant \|\overline{F}\|$. So either $\overline{F}'_1$ or $\overline{F}'_2$ is at least as short as $\overline{F}$, finishing the proof for the left scenario in Fig. 5. The proof for the right scenario in Fig. 5 is analogous, with cases $y \geqslant \sqrt{2}$ and $y \leqslant \sqrt{2}$. This finishes the proof for the right scenario and, hence, for Theorem 2.1.

3 The Tonicity of TSP Tours of Sparse Point Sets

In this section we investigate the tonicity of optimal TSP tours for planar sets P that are sparse. Recall that P is sparse if for any $x \in \mathbb{R}$ the drum $[x, x+1] \times \mathrm{Ball}_{d-1}(\delta/2)$

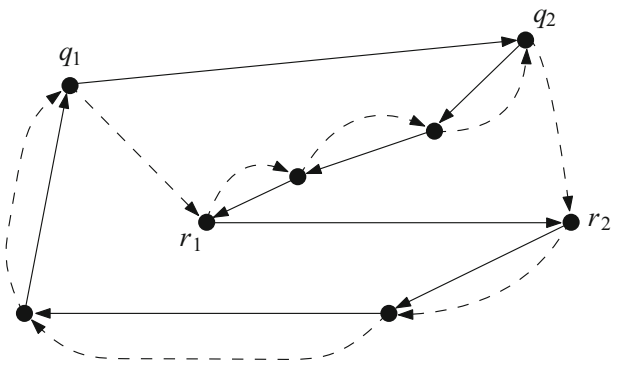


Fig. 6 Illustration of Observation 3.1. T is the solid line, T' is the dashed line. Between $x(r_1)$ and $x(q_2)$, the tonicity of T' is 2 lower than the tonicity of T

contains $O(1)$ points. We start with an easy observation that holds for point sets in an arbitrary number of dimensions, illustrated in Fig. 6.

Observation 3.1 *Let T be a tour containing the (directed) edges q_1q_2 and r_1r_2. Let $|q_1r_1|+|q_2r_2| \leqslant |q_1q_2|+|r_1r_2|$. Then if we swap the edges q_1q_2 and r_1r_2 for the edges q_1r_1 and q_2r_2 we obtain a tour T' with $\|T'\| \leqslant \|T\|$. Similarly, if $|q_1r_1| + |q_2r_2| < |q_1q_2| + |r_1r_2|$ then $\|T'\| < \|T\|$. Furthermore, if q_1 and r_1 are to the left of q_2 and r_2, then $T' \prec T$.*

Proof The fact that $\|T'\| \leqslant \|T\|$ (or $\|T'\| < \|T\|$) is trivial. Furthermore, if q_1 and r_1 are to the left of q_2 and r_2, then $T' \prec T$ holds: between $\max(x(q_1), x(r_1))$ and $\min(x(q_2), x(r_2))$ the tonicity has been lowered by 2. Everywhere else the tonicity remains unchanged, so $T' \prec T$. □

We will need the following observation, illustrated in Fig. 7.

Observation 3.2 *Let $a, b \in \mathbb{R}$. Let q_1, q_2, r_1, r_2 be any four points of P with $\max(x(q_1), x(r_1)) \leqslant a < b \leqslant \min(x(q_2), x(r_2))$ and $|q_1q_2|+|r_1r_2| < |q_1r_1|+|q_2r_2|$. Let q_1', q_2' be the points on q_1q_2 with x-coordinates a, b respectively. Let r_1', r_2' be the points on r_1r_2 with x-coordinates a, b respectively. Then $2(b-a) \leqslant |q_1'q_2'|+|r_1'r_2'| < |q_1'r_1'| + |q_2'r_2'|$.*

Proof When we move q_1 straight towards q_2 until it reaches q_1', then $|q_1r_1|$ cannot decrease more than $|q_1q_2|$. Therefore, $|q_1r_1|-|q_1'r_1| \leqslant |q_1q_2|-|q_1'q_2|$. Combining this with our assumption that $|q_1q_2|+|r_1r_2| < |q_1r_1|+|q_2r_2|$, we get that $|q_1'q_2|+|r_1r_2| < |q_1'r_1| + |q_2r_2|$. Note how we have essentially swapped q_1 for q_1'. We can repeat this process three more times. First we swap q_2 for q_2', then we swap r_1 for r_1', and finally we swap r_2 for r_2'. We thus obtain that $|q_1'q_2'| + |r_1'r_2'| < |q_1'r_1'| + |q_2'r_2'|$. We complete our proof by noting that since $x(q_1') = x(r_1') = a$ and $x(q_2') = x(r_2') = b$, we trivially have that $2(b-a) \leqslant |q_1'q_2'| + |r_1'r_2'|$. □

From now on, we will once more assume that $d = 2$, and that the point set P lies in the δ-strip $\mathbb{R} \times [0, \delta]$. Recall that s_i denotes the separator between points p_i, p_{i+1},

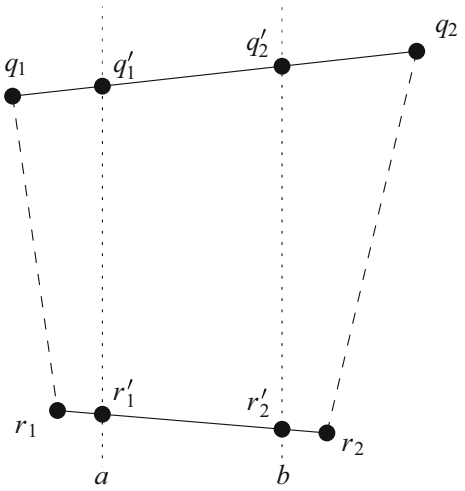


Fig. 7 Illustration of Observation 3.2. Since $|q_1q_2| + |r_1r_2| < |q_1r_1| + |q_2r_2|$ holds, $|q_1'q_2'| + |r_1'r_2'| < |q_1'r_1'| + |q_2'r_2'|$ also holds

and recall the definition $\Delta_i := x_{i+1} - x_i$. A direct consequence of Observation 3.2 is the following.

Lemma 3.3 *Let $k \in \mathbb{N}$ be such that $\delta \leqslant k\Delta_i$. Let T_{opt} be a shortest tour on P. Then there exists a shortest tour $T'_{\text{opt}} \preccurlyeq T_{\text{opt}}$ with $\text{ton}(T'_{\text{opt}}, s_i) \leqslant 2k$.*

Proof Let T_{opt} be a shortest tour. If $\text{ton}(T_{\text{opt}}, s_i) > 2k$, then T_{opt} has at least $k+1$ edges crossing s_i from left to right. We claim that at least one pair of edges (q_1q_2, r_1r_2) has the property that $|q_1q_2| + |r_1r_2| \geqslant |q_1r_1| + |q_2r_2|$. To see this, suppose such a pair does not exist. If we order the edges crossing s_i on where they cross s_i, and then apply Observation 3.2 to each consecutive pair, we get that $2\delta > 2k\Delta_i$. Specifically, let $e_1, ..., e_{k+1}$ be such a set of $k+1$ edges crossing s_i. Let $e_j = (e_{j1}, e_{j2})$ for all j, and let e'_{j1} and e'_{j2} be the points on e_j with x-coordinates x_i and x_{i+1}, respectively. Note that since e_j crosses s_i and is defined by two points of P, e'_{j1} and e'_{j2} are indeed well-defined. Now, from Observation 3.2, we get for all $1 \leqslant j \leqslant k$ that $2(x_{i+1} - x_i) < |e'_{j1}e'_{(j+1)1}| + |e'_{j2}e'_{(j+1)2}|$. See Fig. 8 for an example. Therefore,

$$\begin{aligned} 2k\Delta_i &= \sum_{j=1}^{k} 2(x_{i+1} - x_i) < \sum_{j=1}^{k} |e'_{j1}e'_{(j+1)1}| + |e'_{j2}e'_{(j+1)2}| \\ &= |e'_{11}e'_{(k+1)1}| + |e'_{12}e'_{(k+1)2}| \leqslant 2\delta. \end{aligned}$$

This, however, directly contradicts our assumption that $\delta \leqslant k\Delta_i$. Therefore, such a pair of edges must exist. Using this pair, we can find a tour $T^*_{\text{opt}} \prec T_{\text{opt}}$ with $\|T^*_{\text{opt}}\| \leqslant \|T_{\text{opt}}\|$ (and therefore a shortest tour), by Observation 3.1. We apply this repeatedly until we have obtained a shortest tour $T'_{\text{opt}} \preccurlyeq T_{\text{opt}}$ with $\text{ton}(T'_{\text{opt}}, s_i) \leqslant 2k$. □

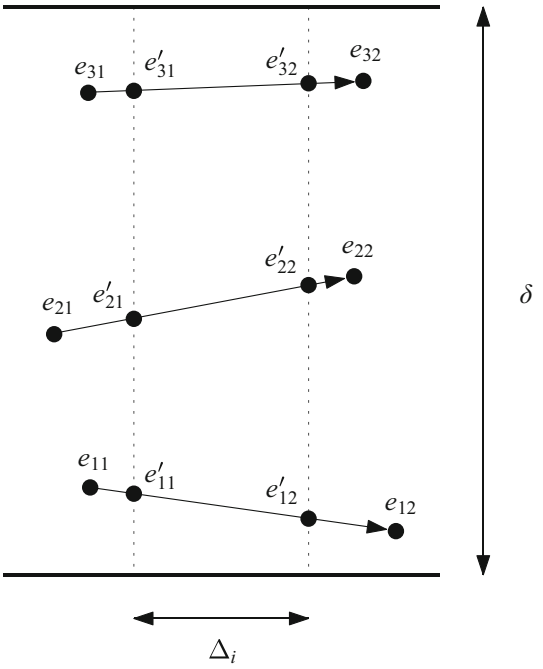


Fig. 8 An example of Lemma 3.3, with $k = 2$. Suppose $|e_1| + |e_2| < |e_{11}e_{21}| + |e_{12}e_{22}|$ and $|e_2| + |e_3| < |e_{21}e_{31}| + |e_{22}e_{32}|$. Since $\delta \geqslant |e'_{11}e'_{21}| + |e'_{21}e'_{31}|$ and $\delta \geqslant |e'_{12}e'_{22}| + |e'_{22}e'_{32}|$, we get that $2\delta \geqslant |e'_{11}e'_{21}| + |e'_{21}e'_{31}| + |e'_{12}e'_{22}| + |e'_{22}e'_{32}| > |e'_{11}e'_{12}| + 2|e'_{21}e'_{22}| + |e'_{31}e'_{32}| \geqslant 4\Delta_i$

A more general version is the following:

Lemma 3.4 *Let $k \in \mathbb{N}$ be such that $\delta \leqslant (x_j - x_i)(k - j + i + 1)$ for some $1 \leqslant i < j \leqslant n$. Let T_{opt} be a shortest tour. Then there exists a shortest tour $T'_{\text{opt}} \preccurlyeq T_{\text{opt}}$ with $\text{ton}(T'_{\text{opt}}, s_i) \leqslant 2k$ and $\text{ton}(T'_{\text{opt}}, s_{j-1}) \leqslant 2k$.*

Proof Let T_{opt} be a shortest tour. If $\text{ton}(T_{\text{opt}}, s_i) > 2k$, then T_{opt} has at least $k+1$ edges crossing s_i from left to right. We claim that at least one pair (q_1q_2, r_1r_2) of crossing edges has the property that $|q_1q_2| + |r_1r_2| \geqslant |q_1r_1| + |q_2r_2|$. To see this, suppose such a pair does not exist. Since at most $j - i - 1$ of these $k + 1$ edges have an endpoint in $\{p_{i+1}, ..., p_{j-1}\}$, at least $k + 1 - (j - i - 1)$ of these have length at least $x_j - x_i$. Analogously to the previous lemma, we order these edges depending on where they cross s_i, and then apply Observation 3.2 to each consecutive pair. This gives us that $2\delta > 2(k - (j - i - 1))(x_j - x_i)$. This, however, directly contradicts our assumption. Therefore, such a pair of edges must exist.

Analogously to the previous lemma, we can use this to find a tour $T_{\text{opt1}} \prec T_{\text{opt}}$ with $\|T_{\text{opt1}}\| \leqslant \|T_{\text{opt}}\|$ (and therefore a shortest tour), by Observation 3.1. We apply this repeatedly until we obtain a shortest tour $T^*_{\text{opt}} \preccurlyeq T_{\text{opt}}$ with $\text{ton}(T^*_{\text{opt}}, s_i) \leqslant 2k$.

By symmetry, we can then find a shortest tour $T'_{\text{opt}} \preccurlyeq T^*_{\text{opt}}$ with $\text{ton}(T'_{\text{opt}}, s_{j-1}) \leqslant 2k$. Since $T'_{\text{opt}} \preccurlyeq T^*_{\text{opt}}$, we also have $\text{ton}(T'_{\text{opt}}, s_i) \leqslant 2k$. Therefore, $T'_{\text{opt}} \preccurlyeq T_{\text{opt}}$ is indeed a shortest tour with $\text{ton}(T'_{\text{opt}}, s_i) \leqslant 2k$ and $\text{ton}(T'_{\text{opt}}, s_{j-1}) \leqslant 2k$. □

We can now bound the tonicity of an optimal TSP tour for P.

Theorem 3.5 *Let P be a set of points inside a δ-strip such that for any $x \in \mathbb{R}$ there are at most c points with x-coordinates in the interval $[x, x+1]$. Then there exists an optimal TSP tour on P that is $2k$-tonic for $k := \lfloor 2\sqrt{c\delta} + 2c \rfloor$.*

Proof Let T_{opt} be a shortest tour. We define $m := \lfloor \sqrt{\delta/c} + 2 \rfloor$. We split the proof into two cases.

- **Case I:** $n < 2cm$. Trivially, T_{opt} is n-tonic. We have

$$n < 2cm = 2c \left\lfloor \sqrt{\delta/c} + 2 \right\rfloor < 2 \left\lfloor 2\sqrt{c\delta} + 2c \right\rfloor = 2k,$$

  so T_{opt} is indeed $2k$-tonic.
- **Case II:** $n \geqslant 2cm$. For any i, j such that $1 \leqslant i$ and $j = i + cm \leqslant n$, we would like to apply Lemma 3.4. To do so, we first argue that $\delta \leqslant (x_j - x_i)(k - j + i + 1)$. Note that since for any $x \in \mathbb{R}$ there are at most c points with x-coordinates in the range $[x, x+1]$, we have $x_j - x_i \geqslant \lfloor (j-i)/c \rfloor$. Therefore, we get

$$\begin{aligned}
&(x_j - x_i)(k - j + i + 1) \\
&\quad \geqslant \left(\left\lfloor \frac{j-i}{c} \right\rfloor - 1 \right) (k + 1 - (j - i)) \\
&\quad = \left(\left\lfloor \sqrt{\delta/c} + 2 \right\rfloor - 1 \right) \left(\left\lfloor 2\sqrt{c\delta} + 2c \right\rfloor + 1 - c \left\lfloor \sqrt{\delta/c} + 2 \right\rfloor \right) \\
&\quad \geqslant \sqrt{\delta/c} \left(2\sqrt{c\delta} + 2c - c \left(\sqrt{\delta/c} + 2 \right) \right) \\
&\quad = \delta.
\end{aligned}$$

  This is true independent of our choice of i. Therefore, we can first apply Lemma 3.4 with $i = 1$, giving us an optimal tour $T_{\text{opt1}} \preccurlyeq T_{\text{opt}}$ that is $2k$-tonic at s_1 and at s_{cm}. When we apply this lemma again on T_{opt1} with $i = 2$, we get a shortest tour $T_{\text{opt2}} \preccurlyeq T_{\text{opt1}}$ that is $2k$-tonic at s_1, s_2, s_{cm} and s_{cm+1}. After doing this for all i such that $i + cm \leqslant n$, we have a shortest tour $T'_{\text{opt}} \preccurlyeq T_{\text{opt}}$ that is $2k$-tonic at all s_i for $1 \leqslant i \leqslant n - cm$ and for $cm \leqslant i \leqslant n$. Since $n \geqslant 2\,cm$, this implies that the tour T'_{opt} is $2k$-tonic at all s_i. □

If the points of point set P have distinct integer x-coordinates (note that we therefore have a sparse point set with $c = 2$), then we can get a slightly better bound in a similar way.

Theorem 3.6 *Let P be a set of points with distinct integer x-coordinates inside a δ-strip. Then there exists an optimal TSP tour on P that is $2k$-tonic for $k := \lfloor 2\sqrt{\delta + 1} \rfloor$.*

Proof The proof is analogous to that of Theorem 3.5, substituting $\lfloor k/2 \rfloor$ for cm. The only significant difference is the proof that $(x_j - x_i)(k - j + i + 1) \geqslant \delta$. This time, we have

$$(x_j - x_i)(k - j + i + 1) \geqslant \left\lfloor \frac{k}{2} \right\rfloor \left(k - \left\lfloor \frac{k}{2} \right\rfloor + 1 \right) = \left\lfloor \frac{k}{2} \right\rfloor \left(\left\lceil \frac{k}{2} \right\rceil + 1 \right).$$

We finish the proof using that $k \in \mathbb{N}$ and $k \geqslant 2\sqrt{\delta+1} - 1$:

$$\left\lfloor \frac{k}{2} \right\rfloor \left(\left\lceil \frac{k}{2} \right\rceil + 1 \right) \geqslant \frac{k-1}{2} \left(\frac{k+1}{2} + 1 \right)$$
$$\geqslant \frac{2\sqrt{\delta+1} - 1 - 1}{2} \left(\frac{2\sqrt{\delta+1} - 1 + 1}{2} + 1 \right) = \delta.$$

□

Finally, we show that the factor $\sqrt{\delta}$ is necessary, by giving a set of examples where the shortest tour has exact tonicity $\Theta(\sqrt{\delta})$:

Theorem 3.7 *For any $k \geqslant 2$, there exists a sparse point set P_k inside a δ-strip of which the shortest tour is unique and has exact tonicity $2k$, with $\delta_k = \Theta(k^2)$.*

Proof Let $\delta_k := 2k^2$. First, we will give a set Q_k of points with distinct integer x-coordinates of which the shortest *path* is unique and consists of the shortest $|Q_k| - 1$ edges. Then, we will add a point t, and argue that the (unique) shortest tour on $P_k = Q_k \cup \{t\}$ is the aforementioned path with both endpoints connected to t.

Let $k \geqslant 2$ be any fixed integer. Let $p := (2k+1, 0)$, let $q_i := (k+i, 2ki)$ and let $r_i = (i, 2ki + k)$. Define

$$Q_k := \{p\} \cup \{q_i \mid i \in \{0, \ldots, k\}\} \cup \{r_i \mid i \in \{0, \ldots, k-1\}\}.$$

See Fig. 9 for an example with $k = 2$. We claim that $\pi := (p, q_0, r_0, q_1, ..., r_{k-1}, q_k)$ is the unique shortest path on Q_k. To see this, first observe that the edge pq_0 and all k edges of the form $q_i r_i$ are the shortest possible edges, with lengths $k+1$ and length $\sqrt{2}k$, respectively. The next shortest are the k edges of the form $r_i q_{i+1}$, with length $\sqrt{2k^2 + 2k + 1}$. Note that these $2k+1$ edges together form the path π. Therefore, if all other possible edges are indeed longer, then π is indeed the shortest path through all points of Q_k, as claimed. The only other serious candidates are of the form $r_i r_{i+1}$ or $q_i q_{i+1}$, all of which have length $\sqrt{4k^2+1}$. Since $k \geqslant 2$, these are indeed longer than the edges in π.

Finally, let t be any point in the δ-strip with a sufficiently large x-coordinate. Then, the two points closest to t are those two that have the highest x-coordinates: p and q_k. Specifically, $t := (3k^4, 2k^2)$ suffices. Since π is a shortest path on Q_k, and starts in p and ends in q_k, we can combine it with the edges $q_k t$ and tp to obtain the shortest tour T through all points of $P_k = Q_k \cup \{t\}$.

In conclusion, for any $k \geqslant 2$, we can find a point set P_k, consisting of $3k^4 + 1$ points with distinct x-coordinates, and a δ_k of $2k^2$, of which the shortest tour is unique and has exact tonicity $2k$. □

4 An Algorithm for Narrow Cylinders

In this section we investigate how the complexity of EUCLIDEAN TSP depends on the width δ of the strip (or cylinder) containing the point set P. Recall that a point set P

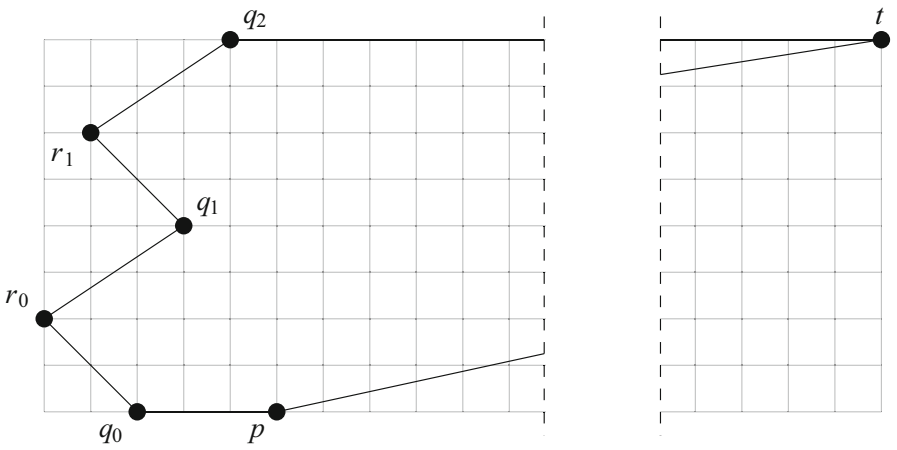


Fig. 9 A sketch of Q_2' and its shortest tour, see Theorem 3.7

inside a δ-cylinder is *sparse* if for every $x \in \mathbb{R}$ the set $[x, x+1] \times \text{Ball}^{d-1}(\delta/2)$ contains $O(1)$ points.

Theorem 4.1 *Let P be a set of n points in a δ-cylinder.*

(i) *If for any $i \in \mathbb{Z}$ the drum $[i\delta, (i+1)\delta] \times \text{Ball}^{d-1}(\delta/2)$ contains at most k points, then we can solve* EUCLIDEAN TSP *on P in $2^{O(k^{1-1/d})}n^2$ time.*
(ii) *If P is sparse then we can solve* EUCLIDEAN TSP *in $2^{O(\delta^{1-1/d})}n^2$ time.*

Part (ii) of the theorem is a trivial consequence of part (i), so the rest of the section focuses on proving part (i). Our proof uses and modifies some techniques of [12]. For $i \in \mathbb{Z}$, let σ_i be the drum $[i\delta, (i+1)\delta] \times \text{Ball}^{d-1}(\delta/2)$. Define $n_i := |\sigma_i \cap P|$—we assume without loss of generality that all points from P lie in the interior of a drum σ_i—and let $k := \max_i n_i$. We say that a drum σ_i is *empty* if $n_i = 0$.

We will regularly use that any subset E of edges from an optimal tour of a point set P has the *Packing Property* [12]: for any $\lambda > 0$ and any cube σ of side length λ, the number of edges from E of length at least $\lambda/4$ that intersect σ is $O(1)$. The Packing Property is at the heart of several subexponential algorithms [21, 29]. We need a recent separator theorem [12, Thm. 5], as explained next.

For a cube σ of side length λ and a number $c > 0$, let $c\sigma$ denote the cube of side length $c\lambda$ and with the same center as σ. We say that an edge e *enters* a cube σ if one endpoint of e is inside σ and the other endpoint is outside σ. The following lemma formalizes the core of the proof of Theorem 3.5 (and Corollary 5.8) in [12]. Theorem 3.5 from that paper is stronger than the lemma below, as it also involves balancing the number of points inside and outside the separator. To guarantee a good balance, a special cube σ is used in the lemma below, but for us that is not relevant.

Lemma 4.2 (De Berg et al. [12]) *Let P be a unknown set of points in $\mathbb{R}^d$. Let Q be a (known) subset of P of m points. Let T be an unknown shortest tour on P, and let E be the (unknown) set of edges of T of which both endpoints are in Q. Then for any cube σ we can in $O(m^{d+1})$ time compute a scale factor $c \in [1, 3]$ such that the*

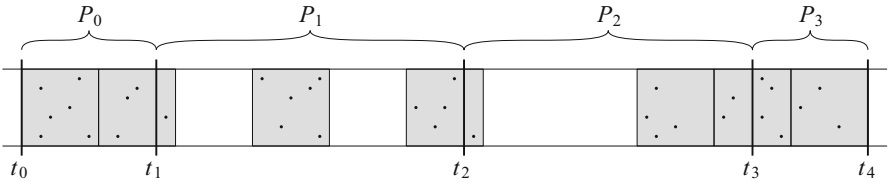


Fig. 10 The separators $t_0, \ldots, t_{|\mathcal{T}|+1}$ and the blocks they define

following holds for the cube $c\sigma$ (which is the cube σ scaled by a factor c with respect to its center).

- $|E_{c\sigma}| = O(m^{1-1/d})$*, where* $E_{c\sigma} \subseteq E$ *denotes the set of edges entering* $c\sigma$.
- *There is a family* $\mathcal{C} \subseteq 2^E$ *of* $2^{O(m^{1-1/d})}$ candidate sets *such that* $E_{c\sigma} \in \mathcal{C}$*, and this family can be computed in* $2^{O(m^{1-1/d})}$ *time.*

Recall that a separator for a set P of points inside a δ-cylinder is a hyperplane orthogonal to the x-axis which does not contain a point from P and which partitions P into two non-empty subsets. Let T_{opt} be an optimal TSP tour on P. For a separator t, let $T(t, \sigma_i)$ denote the set of edges from T_{opt} with both endpoints in $\sigma_{i-1} \cup \sigma_i \cup \sigma_{i+1}$ and crossing t.

Lemma 4.3 *Let σ_i be a drum as defined above. Then we can compute a separator t intersecting σ_i such that $|T(t, \sigma_i)| = O(k^{1-1/d})$ in $O(k^{d+1})$ time. Furthermore, there is a family $\mathcal{C}$ of $2^{O(k^{1-1/d})}$ sets, which we call* candidate sets*, such that $T(t, \sigma_i) \in \mathcal{C}$, and this family can be computed in $2^{O(k^{1-1/d})}$ time.*

Proof We apply Lemma 4.2 to the point set $Q := P \cap (\sigma_{i-1} \cup \sigma_i \cup \sigma_{i+1})$ and the cube σ of side length δ whose left facet contains the right facet of σ_i. The boundary $\partial(c\sigma)$ of the cube $c\sigma$ given by the lemma intersects σ_i (potentially on its boundary), and it is disjoint from the interiors of σ_{i-1} and σ_{i+1}. Let t be the separator containing $\sigma_i \cap \partial(c\sigma)$. The number of points in the three drums is at most $3k$. The number of edges from E crossing t is $O((3k)^{1-1/d}) = O(k^{1-1/d})$. □

Separators and blocks. Consider the sequence of non-empty drums σ_i, ordered from left to right. We use Lemma 4.3 to place a separator in every second drum of this sequence. Let $\mathcal{T} := \{t_1, \ldots, t_{|\mathcal{T}|}\}$ be the resulting (ordered) set of separators. Let t_0 and $t_{|\mathcal{T}|+1}$ denote separators coinciding with the left side of the leftmost non-empty drum and the right side of the rightmost non-empty drum, respectively; see Fig. 10 for an illustration in $\mathbb{R}^2$. We call the region of the δ-cylinder between two consecutive separators t_i and t_{i+1} a *block*. Let $P_i \subseteq P$ denote the set of points in this block. Note that $|P_i| \leqslant 3k$.

For an edge set E and a separator t, let $E(t) \subseteq E$ denote the subset of edges intersecting t. Define $P_{\text{right}}(E, t)$ to be the set of endpoints of the edges in $E(t)$ that lie to the right[1] of t. We call $P_{\text{right}}(E, t)$ the *endpoint configuration of E at t*. The next two lemmas rule out endpoint configurations with two "distant" points from the separator.

[1] The separators given by Lemma 4.3 are not incident to any input points.

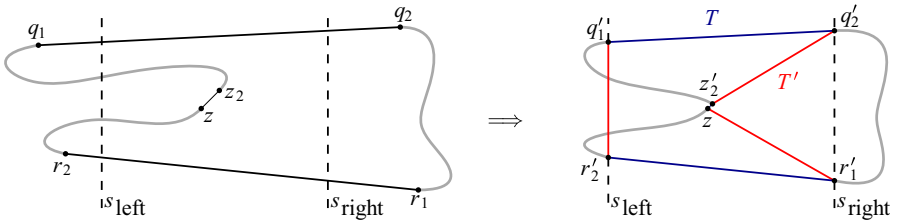


Fig. 11 Illustration for the proof of Lemma 4.4. The point z_2' coincides with z but is slightly displaced for visibility. The sum of the lengths of the edges unique to T (displayed in blue) is strictly larger than the sum of the lengths of the edges unique to T' (displayed in red)

Lemma 4.4 *Let $s_{\text{left}} : x = x_{\text{left}}$ and $s_{\text{right}} : x = x_{\text{right}}$ be two separators such that $x_{\text{right}} - x_{\text{left}} > 3\delta$, and suppose there is a point $z \in P$ with $x_{\text{left}} + 3\delta/2 < x(z) < x_{\text{right}} - 3\delta/2$. Then an optimal tour on P cannot have two edges that both cross s_{left} and s_{right}.*

Proof Suppose for a contradiction that an optimal tour T has two (directed) edges, q_1q_2 and r_1r_2, that both cross s_{left} and s_{right}. (The direction of q_1q_2 and r_1r_2 is according to a fixed traversal of the tour.) If both edges cross s_{left} and s_{right} from left to right (or both cross from right to left) then replacing q_1q_2 and r_1r_2 by q_1r_1 and r_2q_2 gives a shorter tour (see Observation 3.1), leading to the desired contradiction.

Now suppose that q_1q_2 and r_1r_2 cross s_{left} and s_{right} in opposite directions. Assume w.l.o.g. that $x(q_1) < x_{\text{left}}$ and $x(r_2) < x_{\text{left}}$, and that z lies on the path from r_2 to q_1. Let $u_1, \ldots, u_k, v_1, \ldots, v_l, w_1, \ldots, w_m$ and z_2 be such that

$$T = (q_1, q_2, u_1, \ldots, u_k, r_1, r_2, v_1, \ldots, v_l, z, z_2, w_1, \ldots, w_m, q_1).$$

We claim that the tour T' defined as

$$T' = (q_1, r_2, v_1, \ldots, v_l, z, r_1, u_k, \ldots, u_1, q_2, z_2, w_1, \ldots, w_m, q_1)$$

is a strictly shorter tour. To show this, we will first change our point set P into a point set P' such that if $\|T'\| < \|T\|$ on P', then $\|T'\| < \|T\|$ also on P. To this end we replace q_1 by $q_1' := q_1q_2 \cap s_{\text{left}}$ and q_2 by $q_2' := q_1q_2 \cap s_{\text{right}}$, and we replace r_1 by $r_1' := r_1r_2 \cap s_{\text{left}}$ and r_2 by $r_2' := r_1r_2 \cap s_{\text{right}}$; see Fig. 11.

Finally, we replace z_2 by a point z_2' coinciding with z (note that if $z_2 = q_1$, then we can split it before moving the resulting two points, analogous to the proof of Theorem 2.1). Using a similar reasoning as in the proof of Lemma 2.2, one can argue that the point set $P' := (P \setminus \{q_1, q_2, r_1, r_2, z_2\}) \cup \{q_1', q_2', r_1', r_2', z_2'\}$ has the required property. To get the desired contradiction it thus suffices to show that $\|T\| - \|T'\| > 0$ on P'. This is true because

$$\begin{aligned}\|T\| - \|T'\| &= |q_1'q_2'| + |r_1'r_2'| + |zz_2'| - |q_1'r_2'| - |zr_1'| - |q_2'z_2'| \\ &\geqslant |x_{\text{right}} - x_{\text{left}}| + |x_{\text{right}} - x_{\text{left}}| + 0 \\ &\quad -\delta - (|x_{\text{right}} - x(z)| + \delta) - (|x_{\text{right}} - x(z)| + \delta) \\ &> 2(x(z) - x_{\text{left}}) - 3\delta > 0,\end{aligned}$$

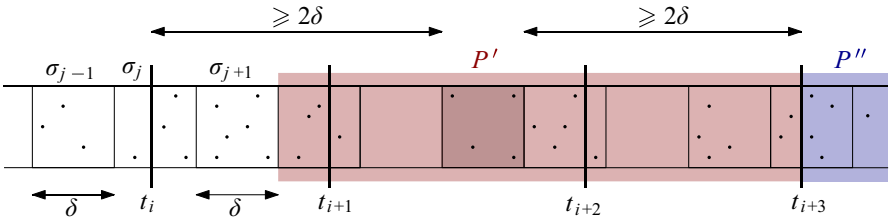


Fig. 12 Only $O(1)$ edges from T_{opt} can cross t_i and the hyperplane $x = (j+2)\delta$, because of the Packing Property. Tour edges crossing t_i and t_{i+3} obey Lemma 4.4. The points in the red area form P'. The points in the blue area form P''

where the last line uses that $x_{\text{left}} + 3\delta/2 < x(z)$. □

Lemma 4.5 *Let $t_i \in \mathcal{T}$ be a separator, and let $\sigma_j = [j\delta, (j+1)\delta] \times \text{Ball}^{d-1}(\delta/2)$ denote the drum in which it is placed. Let T_{opt} be an optimal tour on P and let $V := P_{\text{right}}(T_{\text{opt}}, t_i)$ be its endpoint configuration at t_i. Let P' denote the set of input points with x-coordinates between $(j+2)\delta$ and $x(t_{i+3})$, and let P'' be the set of input points with x-coordinate larger than $x(t_{i+3})$. Then* (i) $|P' \cap V| \leqslant c^*$ *for some absolute constant c^*, and* (ii) $|P'' \cap V| \leqslant 1$.

Proof Consider a tour edge crossing t_i. Any edge crossing t_i ending in P' must fully cross σ_{j+1}, see Fig. 12. Therefore such edges have length at least δ. By the Packing Property, there can be at most $c^* = O(1)$ such edges. This proves (i).

To prove (ii), note that there is a non-empty drum between t_i and t_{i+3} with distance at least 2δ from t_i and t_{i+3}. Lemma 4.4 thus implies that T_{opt} has at most one edge crossing both t_i and t_{i+3}, proving (ii). □

Putting Lemmas 4.3 and 4.5 together, we get the following corollary.

Corollary 4.6 *Let T_{opt} be an optimal tour, let $t_i \in \mathcal{T}$ be a separator, and let $V \subset P$ be the (unknown) endpoint configuration of T_{opt} at t_i. Then we can enumerate in $2^{O(k^{1-1/d})}n$ time a family $\mathcal{B}_i$ of candidate endpoint sets such that $V \in \mathcal{B}_i$.*

Proof Let c^*, P' and P'' be as defined in Lemma 4.5. We use the enumeration of candidate edge sets from Lemma 4.3, and only keep the endpoints that are to the right of t_i. To each of these endpoint sets, we add at most c^* endpoints from P', and we add at most one endpoint from P''. There are $\text{poly}(k)$ and $O(n)$ ways to add such endpoints, respectively. This gives a family of $2^{O(k^{1-1/d})} \cdot \text{poly}(k) \cdot n = 2^{O(k^{1-1/d})}n$ sets. By Lemma 4.5, the resulting family of endpoint sets contains V. □

In addition to the sets $\mathcal{B}_i$ $(i = 1, \ldots, |\mathcal{T}|)$, we define $\mathcal{B}_0 = \mathcal{B}_{|\mathcal{T}|+1} := \emptyset$.

Matchings, the rank-based approach, and representative sets. When we cut a tour using a separator, the tour falls apart into several paths. As in other TSP algorithms, we need to make sure that the paths on each side of the separator can be patched up into a Hamiltonian cycle. Following the terminology of [12], let P be our input point set, and let M be a perfect matching on a set $B \subseteq P$, where the points of B are called

boundary points. A collection $\mathcal{P} = \{\pi_1, \ldots, \pi_{|B|/2}\}$ of paths *realizes* M *on* P if (i) for each edge $(p, q) \in M$ there is a path $\pi_i \in \mathcal{P}$ with p and q as endpoints, and (ii) the paths together visit each point $p \in P$ exactly once. We define the total length of $\mathcal{P}$ as the length of the edges in its paths. In general, the type of problem that needs to be solved on one side of a separator is called EUCLIDEAN PATH COVER. The input to such a problem is a point set $P \subset \mathbb{R}^d$, a set of boundary points $B \subseteq P$, and a perfect matching M on B. The task is to find a collection of paths of minimum total length that realizes M on P.

To get the claimed running time, we need to avoid iterating over all matchings. We can do this with the so-called *rank-based approach* [3, 7]. As our scenario is very similar to the general EUCLIDEAN TSP, we can reuse most definitions and some proof ideas from [12].

Let $\mathcal{M}(B)$ denote the set of all perfect matchings on B, and consider a matching $M \in \mathcal{M}(B)$. We can turn M into a weighted matching by assigning to it the minimum total length of any collection of paths realizing M. In other words, weight(M) is the length of the solution of EUCLIDEAN PATH COVER for input (P, B, M). We use $\mathcal{M}(P, B)$ to denote the set of all such weighted matchings on B. Note that $|\mathcal{M}(P, B)| = |\mathcal{M}(B)| = 2^{O(|B| \log |B|)}$.

We say that two matchings $M, M' \in \mathcal{M}(B)$ *fit* if their union is a Hamiltonian cycle in B. Consider a pair P, B. Let $\mathcal{R}$ be a set of weighted matchings on B and let M be another matching on B. We define $\text{opt}(M, \mathcal{R}) := \min\{\text{weight}(M') : M' \in \mathcal{R}$ and M' fits $M\}$, that is, $\text{opt}(M, \mathcal{R})$ is the minimum total length of any collection of paths on P that together with the matching M forms a cycle. A set $\mathcal{R} \subseteq \mathcal{M}(P, B)$ of weighted matchings is defined to be *representative* of another set $\mathcal{R}' \subseteq \mathcal{M}(P, B)$ if for any matching $M \in \mathcal{M}(B)$ we have $\text{opt}(M, \mathcal{R}) = \text{opt}(M, \mathcal{R}')$. Note that our algorithm is not able to compute a representative set of $\mathcal{M}(P, B)$, because it is also restricted by the Packing Property and Lemma 4.4, while a solution of EUCLIDEAN PATH COVER for a generic P, B, M may not satisfy them. Let $\mathcal{M}^*(P, B)$ denote the set of weighted matchings in $\mathcal{M}(P, B)$ that have a corresponding EUCLIDEAN PATH COVER solution satisfying the Packing Property and Lemma 4.4.

The basis of the rank-based approach is the following result.

Lemma 4.7 (Bodlaender et al. [3, Thm. 3.7]) *There exists a set* $\overline{\mathcal{R}}$ *consisting of* $2^{|B|-1}$ *weighted matchings that is representative of the set* $\mathcal{M}(P, B)$. *Moreover, there is an algorithm* Reduce *that, given a representative set* $\mathcal{R}$ *of* $\mathcal{M}(P, B)$, *computes such a set* $\overline{\mathcal{R}}$ *in* $|\mathcal{R}| \cdot 2^{O(|B|)}$ *time.*

Lemma 4.7 can also be applied in our case, where $\mathcal{R}$ is representative of $\mathcal{M}^*(P, B)$, the set of weighted matchings in $\mathcal{M}(P, B)$ that have a corresponding EUCLIDEAN PATH COVER solution satisfying the Packing Property and Lemma 4.4.

We say that perfect matchings M on B and M' on B' are *compatible* if their union on $B \cup B'$ is either a single cycle or a collection of paths. The *join* of these matchings, denoted by $\text{Join}(M, M')$ is a perfect matching on the symmetric difference $B \triangle B'$ obtained by iteratively contracting edges with an incident vertex of degree 2 in the graph $(B \cup B', M \cup M')$.

The algorithm. Our algorithm is a dynamic program, where we define a subproblem for each separator index i, and each set of endpoints $B \in \mathcal{B}_i$. The value of $A[i, B]$ is defined as follows.

$$A[i, B] := \begin{cases} \text{A representative set containing pairs } (M, x), \text{ where } M \text{ is a perfect} \\ \text{matching on } B \text{ and } x \text{ is a real number equal to the minimum total} \\ \text{length of a path cover of } P_0 \cup \cdots \cup P_{i-1} \cup B \text{ realizing the matching} \\ M. \end{cases}$$

The length of the entire tour will be the value corresponding to the empty matching at index $|\mathcal{T}| + 1$, that is, it will be the value x such that $A[|\mathcal{T}| + 1, \emptyset] = \{(\emptyset, x)\}$.

Our dynamic-programming algorithm works on a block-by-block basis. It uses the algorithm *TSP-repr* by De Berg et al. [12] for EUCLIDEAN PATH COVER on arbitrary d-dimensional point sets to solve subproblems inside a block. In particular, *TSP-repr* computes a set of weighted matchings that represents $\mathcal{M}^*(P, B)$. Algorithm 1 gives our algorithm in a pseudocode, which is further explained below.

Algorithm 1 NarrowRectTSP-DP(P, δ)

Input: A set P of points in $(-\infty, \infty) \times \text{Ball}^{d-1}(\delta/2)$ that is sparse
Output: The length of the shortest tour through all points in P

1: Compute the separators $t_0, \ldots, t_{|\mathcal{T}|+1}$ using Lemma 4.3, as explained above.
2: Compute the sets $\mathcal{B}_0, \ldots, \mathcal{B}_{|\mathcal{T}|+1}$ as explained in the proof of Corollary 4.6.
3: $A[0, \emptyset] := \{(\emptyset, 0)\}$
4: **for** $i = 1$ to $|\mathcal{T}| + 1$ **do**
5: **for all** $B \in \mathcal{B}_i$ **do**
6: $A[i, B] := \emptyset$
7: **for all** $B' \in \mathcal{B}_{i-1}$ where $B' \subseteq P_{i-1} \cup B$ **do**
8: **for all** $(M, x) \in$ *TSP-repr*$(P_{i-1} \cup B,\ B' \triangle B)$ **do**
9: **for all** $(M', x') \in A[i-1, B']$ **do**
10: **if** M' and M are compatible **then**
11: Insert (Join$(M, M'), x + x'$) into $A[i, B]$
12: **end if**
13: **end for**
14: **end for**
15: **end for**
16: *Reduce*$(A[i, B])$
17: **end for**
18: **end for**
19: **return** length$(A[|\mathcal{T}| + 1, \emptyset])$

The goal of Lines 5–16 is to compute a representative set $A[i, B]$ of $\mathcal{M}^*(P_0 \cup \cdots \cup P_{i-1} \cup B,\ B)$ of size $2^{O(k^{1-1/d})}$. We say that a point $p \in P$ is *distant* (with respect to a separator t_i) if it is to the right of t_{i+3}, and denote the set of distant input points from t_i by distant(t_i). First, we iterate over all sets $B \in \mathcal{B}_i$ in Line 5. Next, we consider certain boundary sets $B' \in \mathcal{B}_{i-1}$. Note that if there is a distant point $p \in B' \cap$ distant(t_i), then a tour edge crossing t_{i-1} ending at p also crosses t_i, and thus p is also a (distant) point of B.

In Line 8 we call the algorithm of De Berg et al. [12] on the point set $P_{i-1} \cup B' \cup B$ with $B' \triangle B$ as boundary set; note that $P_{i-1} \cup B' \cup B = P_{i-1} \cup B$ (since $B' \subseteq P_{i-1} \cup B$),

so the first parameter of the call can be written as $P_{i-1} \cup B$. This gives us a representative set $\mathcal{R}$ of $\mathcal{M}^*(P_{i-1} \cup B,\ B' \triangle B)$. For each weighted matching $(M, x) \in \mathcal{R}$, and for each weighted matching from the representative set $(M', x') \in A[i-1, B']$, we check whether M and M' are compatible. If so, then taking the union of the corresponding path covers gives a path cover of $P_0 \cup \cdots \cup P_{i-1} \cup B$ of total length $x + x'$, which realizes the matching $\text{Join}(M, M')$ on B; we then add $(\text{Join}(M, M^*), x + x^*)$ to $A[i, B]$ in Line 11.

After iterating over all boundary sets B', the entry $A[i, B]$ stores a set of weighted matchings. These weighted matchings will be used on the next iteration at Line 9 to generate the $A[i+1, \cdot]$ entries. To prevent each iteration taking more time than the previous one, thus blowing up the constants, we reduce $A[i, B]$ to size at most $2^{Mk^{1-1/d}}$ for some large but fixed M using the *Reduce* algorithm [3] in Line 16.

Note that in the final iteration, when $i = t+1$, we take $B = \emptyset$. Now M and M' are compatible if and only if the union of the corresponding path covers is a Hamiltonian cycle. Line 16 then gives to a single entry the smallest weight. Therefore, the length of the only entry in $A[t+1, \emptyset]$ after the loops have ended, is the length of the optimal TSP tour. Hence, the correctness of NarrowRectTSP-DP follows from the next lemma.

Lemma 4.8 *After Step 16, the set $A[i, B]$ is a representative set of $\mathcal{M}^*(P_0 \cup \cdots \cup P_{i-1} \cup B, B)$.*

Proof This proof is very similar to the correctness proof of [12]. We use induction on i. For $i = 1$, we are directly calling *TSP-repr*$(P_0 \cup B, B)$, which gives a representative set. This set is reduced to a potentially smaller set on Line 16, which is still a representative set by Lemma 4.7.

Assume now that $i > 1$, and for each $B' \in \mathcal{B}_{i-1}$ the family $A[i-1, B']$ is representative of $\mathcal{M}^*(P_0 \cup \cdots \cup P_{i-2} \cup B', B')$. For any fixed $B' \in \mathcal{B}_{i-1}$, we have that the set returned by *TSP-repr* is a representative set of $\mathcal{M}^*(P_{i-1} \cup B,\ B' \triangle B)$. By Lemma 3.6 in [3], the join operation executed on all compatible pairs of weighted matchings from two representative sets results in a representative set, thus for each fixed B', we have inserted a representative set of

$$\mathcal{M}_{B'} := \{\text{Join}(M', M) \mid M' \in \mathcal{M}^*(P_0 \cup \cdots \cup P_{i-2} \cup B',\ B'),\\ M \in \mathcal{M}^*(P_{i-1} \cup B,\ B' \triangle B)\}$$

into $A[i, B]$. Consequently, just before executing Line 16 the set $A[i, B]$ is a representative set of $\bigcup_{B' \in \mathcal{B}'} \mathcal{M}_{B'}$, where $\mathcal{B}'$ is the family of endpoint configurations we consider at Line 7. Note that $\mathcal{B}'$ is a (super)set of all possible endpoint configurations in $\mathcal{B}_{i-1}$ that a path cover of $P_0 \cup \cdots \cup P_{i-1} \cup B$ with boundary B obeying the Packing Property and Lemma 4.4 can have. The set $\mathcal{M}_{B'}$ contains the subset of $\mathcal{M}^*(P_0 \cup \cdots \cup P_{i-1}, B)$ corresponding to path covers where the endpoint configuration at t_{i-1} is B'. Consequently, $\mathcal{M}^*(P_0 \cup \cdots \cup P_{i-1}, B) \subseteq A[i, B]$. Therefore $A[i, B]$ is a representative set of $\mathcal{M}^*(P_0 \cup \cdots \cup P_{i-1}, B)$ just before executing Line 16, and by Lemma 4.7, it remains a representative set after executing this line. □

Analysis of the running time. The loop of Lines 4–16 has $|\mathcal{T}| + 1 = O(n)$ iterations. Each set $\mathcal{B}_i$ contains $2^{O(k^{1-1/d})}n$ sets. For each choice of $B \in \mathcal{B}_i$, we have $2^{O(k^{1-1/d})}$

options for B', since B can have at most one point distant from t_i by Lemma 4.4. The running time of *TSP-repr* is $T(|P|, |B|) = 2^{O(|P|^{1-1/d}+|B|)}$ [12, Lem. 8]. By Lemma 4.3 we have $|B| = O(k^{1-1/d})$, so the running time of each call to *TSP-repr* in Algorithm 1 is

$$2^{O((3k+|B|)^{1-1/d}+|B|^{1-1/d})} = 2^{O(k^{1-1/d})}.$$

The representative set returned by *TSP-repr* has $2^{O(k^{1-1/d})}$ weighted matchings, and the representative set of $A[i-1, B']$ also has $2^{O(k^{1-1/d})}$ matchings. Checking compatibility, joining and insertion in Lines 10 and 11 takes $\text{poly}(|M|, |M'|) = \text{poly}(k)$ time. Consequently, before executing the reduction in Line 16, the set $A[i, B]$ contains at most $2^{O(k^{1-1/d})} \cdot 2^{O(k^{1-1/d})} = 2^{O(k^{1-1/d})}$ entries. By Lemma 4.7 the application of the *Reduce* algorithm results in a representative set of size at most $2^{|B|-1} = 2^{O(k^{1-1/d})}$. Hence, the total running time is

$$n \cdot 2^{O(k^{1-1/d})} n \cdot 2^{O(k^{1-1/d})} \cdot \left(2^{O(k^{1-1/d})} + 2^{O(k^{1-1/d})} poly(k)\right) = 2^{O(k^{1-1/d})} n^2.$$

This concludes the proof of Theorem 4.1.

5 Random Point Sets Inside a Narrow Rectangle

Algorithm 1 also works efficiently on random point sets, with only minimal changes. Specifically, in this section we will prove the following.

Theorem 5.1 *Let P be a set of n points taken independently uniformly at random from the hypercylinder $[0, n] \times \text{Ball}^{d-1}(\delta/2)$. Then we can solve* Euclidean TSP *on P in $2^{O(\delta^{1-1/d})} n$ expected time.*

Our proof has four steps. In the first step, we will show that long edges are unlikely to be viable. For the second step, recall the definition of the *spacing* of p_i (in P) as $\Delta_i = x_{i+1} - x_i$, for all $1 \leqslant i \leqslant n-1$. We extend this definition with $\Delta_0 = x_1$, and $\Delta_n = n - x_n$. Note that $\sum_{i=0}^{n} \Delta_i = n$. The values Δ_i play an important role in the analysis of the algorithm, and it will be convenient to assume that the Δ_i are independent. However, when the x-coordinates of the points are generated uniformly at random in the interval $[0, n]$, this is not quite true. In the second step, we therefore describe a method to generate the random point set in a different way, and we show how to relate the expected running times in these two settings. In the third step, we will explain which changes are made to the algorithm. Finally, in the fourth step, we will use the previous steps to analyse the expected running time of the algorithm.

Step 1: Long edges are unlikely to be viable. In this step, we will create a pattern of points through which no long edge can pass. To do so, we will first show that if a long edge is in an optimal tour T, then T must be bitonic at certain separators. Recall that s_j is the separator between points p_j and p_{j+1}.

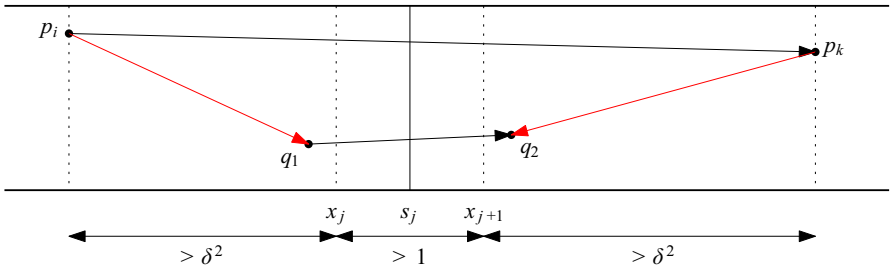


Fig. 13 An example of Observation 5.2. Edges of T are shown in black, edges of T' are shown in red

Observation 5.2 *Let $i < j < k$ with $x_i + \delta^2 < x_j$ and $\Delta_j > 1$ and $x_{j+1} + \delta^2 < x_k$. Let T be an optimal tour on P, and suppose that T contains $p_i p_k$. Then T is bitonic at s_j.*

Proof See Fig. 13 for an example. W.l.o.g., T contains the directed edge from p_i to p_k. Suppose for a contradiction that T is an optimal tour and not bitonic at s_j. Then T must cross s_j at least four times, which implies that there must be another directed edge q_1q_2 with $x(q_1) \leqslant x_j$ and $x_{j+1} \leqslant x(q_2)$. Note that $q_1 \neq p_i$ and $q_2 \neq p_k$, since a single point cannot have two incoming or two outgoing edges. We will now show that $|p_iq_1| + |p_kq_2| < |q_1q_2| + |p_ip_k|$. By Observation 3.1, we then get an optimal tour T' shorter than T, thereby finishing the proof. First, we will simplify the problem using an argument similar to the proof of Observation 3.2. Suppose we move q_1 towards q_2 until $x(q_1) = x_j$. By doing so, $|q_1p_i|$ will never decrease more than $|q_1q_2|$. Therefore, it is sufficient to prove the statement for $x(q_1) = x_j$. Analogously, it is sufficient to prove the statement for $x(q_2) = x_j + 1$ and $x_i = x_j - \delta^2$ and $x_k = x_j + 1 + \delta^2$. Together, we now have

$$x_i + \delta^2 = x(q_1) = x(q_2) - 1 = x_k - 1 - \delta^2.$$

We get

$$\begin{aligned}
|p_iq_1| + |p_kq_2| &\leqslant \sqrt{(x(q_1) - x_i)^2 + \delta^2} + \sqrt{(x(q_2) - x_k)^2 + \delta^2} \\
&= \sqrt{(\delta^2)^2 + \delta^2} + \sqrt{(\delta^2)^2 + \delta^2} \\
&= 2\sqrt{\delta^4 + \delta^2} \\
&< 2\sqrt{\delta^4 + 2\delta^2 + 1} \\
&= 2\delta^2 + 2 \\
&= 1 + (2\delta^2 + 1) \\
&= (x(q_2) - x(q_1)) + (x_k - x_i) \\
&\leqslant |q_1q_2| + |p_ip_k|.
\end{aligned}$$

By Observation 3.1, this gives us an optimal tour T' shorter than T, thereby finishing the proof. □

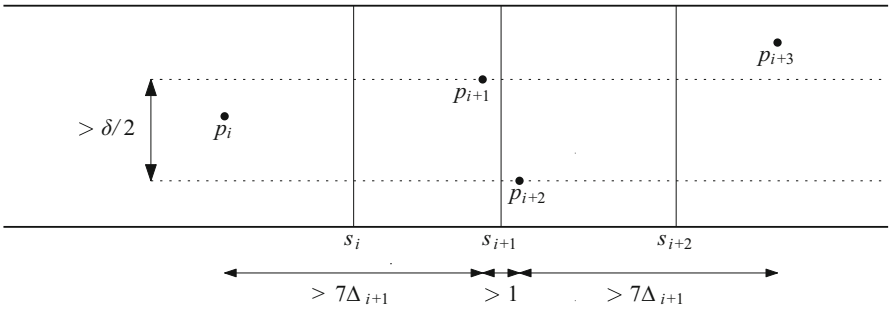


Fig. 14 An example of a wall, for $d = 2$

We are now ready to create our pattern. We define a *wall* to be a set of four points $(p_i, \ldots, p_{i+3})$ with the following properties:

- $\Delta_i > 7\Delta_{i+1}$ and $\Delta_{i+1} > 1$ and $\Delta_{i+2} > 7\Delta_{i+1}$, and
- $|p'_{i+1}p'_{i+2}| > \delta/2$,

where p'_{i+1} and p'_{i+2} are the orthogonal projections of p_{i+1} and p_{i+2} onto the ball $\{0\} \times \text{Ball}^{d-1}(\delta/2)$, respectively. See Fig. 14 for an example of a wall. Now we will prove that walls indeed tell us something about the validity of long edges. To do so, we will need the following lemma:

Lemma 5.3 *Let $a, b, c > 0$. If $4ab - (4c - a - b)^2 \geqslant 0$, then $\sqrt{a} + \sqrt{b} - 2\sqrt{c} \geqslant 0$.*

Proof We split the proof into two cases:

- **Case** $4c - a - b \geqslant 0$. We get

$$\begin{aligned}
\sqrt{a} + \sqrt{b} &= \sqrt{a + b + \sqrt{4ab}} \\
&\geqslant \sqrt{a + b + \sqrt{(4c - a - b)^2}} && \text{since } 4ab - (4c - a - b)^2 \geqslant 0 \\
&= \sqrt{a + b + 4c - a - b} && \text{since } 4c - a - b \geqslant 0 \\
&= 2\sqrt{c}.
\end{aligned}$$

- **Case** $4c - a - b < 0$. We get

$$\begin{aligned}
\sqrt{a} + \sqrt{b} &= \sqrt{a + b + \sqrt{4ab}} \\
&> \sqrt{a + b} \\
&> 2\sqrt{c} && \text{since } 4c - a - b < 0.
\end{aligned}$$

This concludes the proof of Lemma 5.3. □

We are now ready to prove that long edges passing through the wall must have one endpoint close to the wall.

Lemma 5.4 *Let $(p_i, \ldots, p_{i+3})$ be a wall. Let q_1q_2 be an edge with $x(q_1) + \delta^2 < x_i$ and $x_{i+3} + \delta^2 < x(q_2)$. Then the shortest tour on the points of P does not contain q_1q_2.*

Proof Suppose for a contradiction that T is an optimal tour that contains w.l.o.g. the directed edge q_1q_2. By Observation 5.2, T is bitonic at s_i and s_{i+1} and s_{i+2}. Note that q_1q_2 passes through all three of these separators. Therefore, one of the neighbours of p_{i+1} must be to the left of p_{i+1}, and one must be to its right. The same holds for p_{i+2}. Since both q_1q_2 and the edge from p_{i+2} to its 'right' neighbour pass through s_{i+2}, the edge from p_{i+1} to its 'right' neighbour cannot pass through s_{i+2}. Therefore, the 'right' neighbour of p_{i+1} is p_{i+2}. In conclusion, T must contain the directed edge $p_{i+2}p_{i+1}$.

It is sufficient to show that $|q_1p_{i+2}| + |q_2p_{i+1}| < |q_1q_2| + |p_{i+2}p_{i+1}|$, because this will contradict the optimality of T by Obervation 3.1. Analogously to the proof of Observation 3.2, it is sufficient to prove this for $x(q_1) + \delta^2 + 7\Delta_{i+1} = x_{i+1}$ and $x_{i+2} + \delta^2 + 7\Delta_{i+1} = x(q_2)$. Combining these simplifications, we want to prove that

$$\begin{aligned}
&|q_1q_2| + |p_{i+2}p_{i+1}| - |q_1p_{i+2}| - |q_2p_{i+1}| \\
&\quad > \sqrt{(2\delta^2 + 15\Delta_{i+1})^2 + 0^2} + \sqrt{\Delta_{i+1}^2 + \delta^2/4} - 2\sqrt{(\delta^2 + 8\Delta_{i+1})^2 + \delta^2} \\
&\quad \geqslant 0.
\end{aligned}$$

We invoke Lemma 5.3 with $a = (2\delta^2 + 15\Delta_{i+1})^2$ and $b = \Delta_{i+1}^2 + \delta^2/4$ and $c = (\delta^2 + 8\Delta_{i+1})^2 + \delta^2$. We check whether the requirement holds. A bit of algebra gives us that for these values we can rewrite $4ab - (4c - a - b)^2$ to

$$\left(30\Delta_{i+1} - \frac{225}{16}\right)\delta^4 + 4\delta^6,$$

which is indeed at least 0 for all $\delta > 0$ and $\Delta_{i+1} > 1$, as required.

Therefore, we have now proven that $|q_1q_2|+|p_{i+2}p_{i+1}|-|q_1p_{i+2}|+|q_2p_{i+1}| > 0$. Therefore, T is not an optimal tour, leading to the conclusion that a shortest tour on the points of P does not contain q_1q_2. □

Note that if the Δ_i are i.i.d. (independent and identically distributed) and the other coordinates of the points are taken independently uniformly at random from $\text{Ball}^{d-1}(\delta/2)$, there exists some constant probability that $(p_i, \ldots, p_{i+3})$ is a wall. This probability is independent of δ, n, i, and whether any $(p_j, \ldots, p_{j+3})$ is a wall for any $j \leqslant i - 3$ or $j \geqslant i + 3$. We will expand on this later. This brings us to our next step.

Step 2: Another way of generating points, with independent Δ_i. As n goes to infinity, the Δ_i become more and more independent. True independence would greatly simplify the calculation of the running time. Therefore, let us take a look at a different way of generating our point set.

First, we generate $n+1$ independent exponentially distributed variables, $\Delta_0, \ldots, \Delta_n \sim \text{Exp}(1)$. Using these, we compute the x-coordinates of our points, defined as

$x_i := \sum_{j=0}^{i-1} \Delta_i$ for $1 \leqslant i \leqslant n$. We will also write $x_{n+1} := \sum_{i=0}^{n} \Delta_i$ for brevity. Note that the value of x_{n+1} will very likely be relatively close to n, since its distribution converges to a normal distribution with mean $n+1$ and standard deviation $\sqrt{n+1}$ when $n \to \infty$. Finally, we generate the other coordinates by picking n independent uniformly distributed points in $\text{Ball}^{d-1}(\delta/2)$. Combining these gives us n points in the hypercylinder $[0, x_{n+1}] \times \text{Ball}^{d-1}(\delta/2)$.

A point set generated this way has two important properties. First of all, the Δ_i are now independent. Second, as we will prove next, the expected running time of an algorithm on a uniformly distributed point set can be bounded by the expected running time of that algorithm on a point set generated this way. We can even generalize this for more arbitrary point set distributions.

Let X_n be a random point set of n points in $\mathbb{R}^d$, where the x-coordinates of the points are taken independently uniformly at random from the interval $[0, n]$, and the other coordinates are i.i.d. and independent of the x-coordinates. Let Y_n be a random point set of n points in $\mathbb{R}^d$, where the spacings $\Delta_i = x_{i+1} - x_i$ are taken independently from Exp(1) (including Δ_n to create an x_{n+1}), and the other coordinates are i.i.d. and independent of the x-coordinates, with the same distribution as in X_n. We then get

Lemma 5.5 *Let X_n and Y_n be as defined above. Let $\mathcal{A}$ be an algorithm that runs on point sets in $\mathbb{R}^d$. Let $T_{\mathcal{A}}(P)$ denote its running time for a point set P. Suppose for all $\lambda > 1$ and P, we have $T_{\mathcal{A}}(P_\lambda) \leqslant T_{\mathcal{A}}(P)$, where P_λ is obtained from P by scaling the x-coordinates by a factor $\lambda > 1$. Suppose $\mathbb{E}[T_{\mathcal{A}}(Y_n)] = f(n)$ for some function f. Then $\mathbb{E}[T_{\mathcal{A}}(X_n)] = O(f(n))$.*

Proof First, we note that X_n and Y_n are remarkably similar: when the x-coordinates of the points of Y_n are scaled such that $x_{n+1} = n$, we obtain the same distribution as X_n, see [8]. Therefore,

$$\mathbb{E}[T_{\mathcal{A}}(X_n)] = \mathbb{E}[T_{\mathcal{A}}(Y_n) | x_{n+1} = n].$$

Since $T_{\mathcal{A}}(P_\lambda) \leqslant T_{\mathcal{A}}(P)$ for all $\lambda > 1$, P, we get that

$$\mathbb{E}[T_{\mathcal{A}}(Y_n) | x_{n+1} = n] \leqslant \mathbb{E}[T_{\mathcal{A}}(Y_n) | x_{n+1} \leqslant n].$$

Finally, since the distribution of x_{n+1} converges to a normal distribution with mean $n+1$ and standard deviation $\sqrt{n+1}$ when $n \to \infty$, we have $\mathbb{P}[x_{n+1} \leqslant n] \to \frac{1}{2}$ when $n \to \infty$. Therefore,

$$\begin{aligned}
\mathbb{E}[T_{\mathcal{A}}(Y_n) | x_{n+1} \leqslant n] &< 3 \cdot \mathbb{P}[x_{n+1} \leqslant n] \cdot \mathbb{E}[T_{\mathcal{A}}(Y_n) | x_{n+1} \leqslant n] \text{for } n \text{ large enough} \\
&\leqslant 3 \cdot \mathbb{P}[x_{n+1} \leqslant n] \cdot \mathbb{E}[T_{\mathcal{A}}(Y_n) | x_{n+1} \leqslant n] \\
&\quad + 3 \cdot \mathbb{P}[x_{n+1} > n] \cdot \mathbb{E}[T_{\mathcal{A}}(Y_n) | x_{n+1} > n] \\
&= 3 \cdot \mathbb{E}[T_{\mathcal{A}}(Y_n)] \\
&= 3 \cdot f(n) = O(f(n)),
\end{aligned}$$

finishing the proof. □

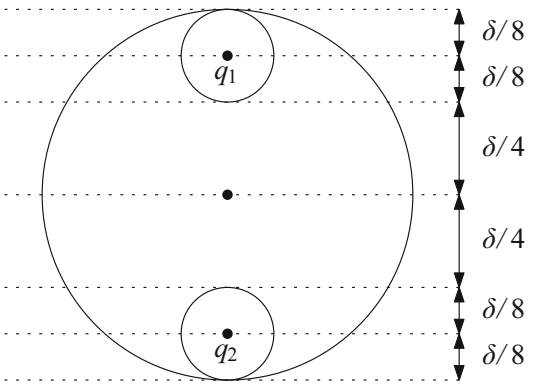


Fig. 15 An example with $d = 3$. The large circle is the separator s_{i+1}. If p'_{i+1} is in the upper circle $\{0\} \times B^{d-1}(q_1, \delta/8)$, and p'_{i+2} is in the lower circle $\{0\} \times B^{d-1}(q_2, \delta/8)$, then $|p'_{i+1}p'_{i+2}|$ is at least $\delta/2$. Note that the corresponding probabilities are equal, and that the location of p'_{i+1} is independent of the location of p'_{i+2}

Now we can give a simple lower bound on the probability that $(p_i, \ldots, p_{i+3})$ forms a wall. Let $B^{d-1}(p, r)$ be the $(d-1)$-dimensional ball with midpoint p and radius r. Define q_1 as $x(q_1) = 0, y_1(q_1) = 3\delta/8, y_2(q_1) = \ldots = y_{d-2}(q_1) = 0$, and define q_2 as $x(q_2) = 0, y_1(q_2) = -3\delta/8, y_2(q_2) = \ldots = y_{d-2}(q_2) = 0$. Recall that p'_{i+1} and p'_{i+2} are the orthogonal projections of p_{i+1} and p_{i+2}, respectively, on $\{0\} \times \text{Ball}^{d-1}(\delta/2)$. Now, note that if $p'_{i+1} \in \{0\} \times B^{d-1}(q_1, \delta/8)$ and $p'_{i+2} \in \{0\} \times B^{d-1}(q_2, \delta/8)$, then $|p'_{i+1}p'_{i+2}| \geqslant |q_1q_2| - 2\delta/8 = \delta/2$. Then

$$
\begin{aligned}
&\mathbb{P}[\Delta_i > 7\Delta_{i+1} \wedge \Delta_{i+1} > 1 \wedge \Delta_{i+2} > 7\Delta_{i+1} \wedge |p'_{i+1}p'_{i+2}| > \delta/2] \\
&\quad > \mathbb{P}[\Delta_i > 14 \wedge 1 < \Delta_{i+1} < 2 \wedge \Delta_{i+2} > 14 \wedge |p'_{i+1}p'_{i+2}| > \delta/2] \\
&\quad = e^{-14} \cdot (e^{-1} - e^{-2}) \cdot e^{-14} \cdot \mathbb{P}[|p'_{i+1}p'_{i+2}| > \delta/2] \quad \text{since all } \Delta_j \sim \text{Exp}(1) \\
&\quad = \Theta(1) \cdot \mathbb{P}[|p'_{i+1}p'_{i+2}| > \delta/2] \\
&\quad > \Theta(1) \cdot \mathbb{P}[p'_{i+1} \in \{0\} \times B^{d-1}(q_1, \delta/8)]^2 \quad \text{see Fig. 15.}
\end{aligned}
$$

Now, it is well-known that the volume of the $(d-1)$-dimensional ball with radius r is $c_d r^{d-1}$ for some constant c_d. Since p'_{i+1} is a random point uniformly at random in $\{0\} \times \text{Ball}^{d-1}(\delta/2)$, we get

$$
\begin{aligned}
&\Theta(1) \cdot \mathbb{P}[p'_{i+1} \in \{0\} \times B^{d-1}(q_1, \delta/8)]^2 \\
&\quad = \Theta(1) \cdot \left(\frac{c_d(\delta/8)^{d-1}}{c_d(\delta/2)^{d-1}} \right)^2 \\
&\quad = \Theta(1) \cdot \left(2^{-2(d-1)} \right)^2 \\
&\quad = \Theta(1),
\end{aligned}
$$

since we assume d to be a constant.

Step 3: A new algorithm. The new algorithm is exactly the same as the old algorithm, with three changes. The first change is simple: we previously stated that we start by sorting our points on their x-coordinate. In the case of sparse point sets we use a worst-case $O(n \log n)$ sorting algorithm for this, but now we can use a bucket sort with n buckets. This brings the expected sorting time to $O(n)$ [5].

For the second change, we need to take another look at how we place the separators t_i. We previously placed these separators in every second nonempty drum $\sigma_i := [i\delta, (i+1)\delta] \times \text{Ball}^{d-1}(\delta/2)$ based on the points in $\sigma_{i-1} \cup \sigma_i \cup \sigma_{i+1}$. However, in order for our algorithm to meet the requirements of Lemma 5.5, we would like to avoid having a point enter a drum after the x-coordinates are multiplied by some factor $\lambda > 1$. Furthermore, since the proof of Lemma 4.3 requires every drum to be at least δ wide, we cannot simply scale the drums as well. Therefore, we will need to allow some overlap. For $j \in \{-1, 0, 1\}$ and all i, we define $\tau_{i,j} := [(X(i) - 1/2 + j)\delta, (X(i) + 1/2 + j)\delta] \times \text{Ball}^{d-1}(\delta/2)$, with $X(i) := (i + 1/2)x_{n+1}/n$. Note that the center of drum $\tau_{i,0}$ scales with the point set, and $\tau_{i,-1}$ and $\tau_{i,1}$ are the drums directly to its left and right. In our new algorithm, we will use the drums $\tau_{i,j}$ instead of the σ_i used so far. Since this is quite a drastic change, before we show how exactly these new drums are used in the algorithm, we will need to revisit all drum-related lemmas and corollaries. Let us start with Lemma 4.3. When applied to the new drums, it gives us the following.

Lemma 5.6 *Let T_{opt} be an optimal TSP tour on P. For a separator t, let $T(t, \tau_{i,0})$ denote the set of edges from T_{opt} with both endpoints in $\tau_{i,-1} \cup \tau_{i,0} \cup \tau_{i,1}$ and crossing t. Let k be the number of points in $\tau_{i,-1} \cup \tau_{i,0} \cup \tau_{i,1}$. Then in $O(k^{d+1})$ time we can compute a separator t intersecting $\tau_{i,0}$ such that $|T(t, \tau_{i,0})| = O(k^{1-1/d})$. Furthermore, there is a family $\mathcal{C}$ of $2^{O(k^{1-1/d})}$ candidate sets such that $T(t, \tau_{i,0}) \in \mathcal{C}$, and this family can be computed in $2^{O(k^{1-1/d})}$ time.*

Proof The proof is analogous to that of Lemma 4.3. □

Note that k is now defined 'locally', since we no longer have a good bound on the maximum density of our point set. Instead of placing the separators t_i in every second nonempty σ_i, we place them in every $\tau_{i,0}$ instead, using the points in $\tau_{i,-1} \cup \tau_{i,0} \cup \tau_{i,1}$. We continue to Lemma 4.5, since Lemma 4.4 is independent of the drums used. We first define $P[i, j] := \{p_i, \ldots, p_j\}$ for all $1 \leqslant i \leqslant j \leqslant n$. The revised version of Lemma 4.5 is the following:

Lemma 5.7 *Let $t_i \in \mathcal{T}$ be a separator. Let T_{opt} be an optimal tour on P and let $V := P_{\text{right}}(T_{\text{opt}}, t_i)$ be its endpoint configuration at t_i. Let z be the leftmost point with an x-coordinate larger than $(X(i)+3)\delta$. Let p_j be the leftmost point more than $3\delta/2$ to the right of z. Let P' denote the set of points with x-coordinates between $(X(i)+3/2)\delta$ (the right border of $\tau_{i,1}$) and x_j, and let $P'' = P[j, n]$. Then* (i) *$|P' \cap V| \leqslant c^*$ for some absolute constant c^*, and* (ii) *$|P'' \cap V| \leqslant 1$.*

Proof Consider a tour edge crossing t_i. Any edge crossing t_i ending in P' must fully cross $\tau_{i,1}$. Therefore such edges have length at least δ. By the Packing Property, there can be at most $c^* = O(1)$ such edges. This proves (i).

To prove (ii), we can directly apply Lemma 4.4. Note that point z indeed suffices, since it is more than $3\delta/2$ to the right of the right border of $\tau_{i,1}$. □

Next up is Corollary 4.6, which we adapt as follows:

Corollary 5.8 *Let T_{opt} be an optimal tour. Let $t_i \in \mathcal{T}$ be a separator. Let $V \subset P$ be the (unknown) endpoint configuration of T_{opt} at t_i. Then in $2^{O(k^{1-1/d})}n$ time we can enumerate a family $\mathcal{B}_i$ of candidate endpoint sets such that $V \in \mathcal{B}_i$. Here, k denotes the number of points which either lie in $\tau_{i,-1} \cup \tau_{i,0} \cup \tau_{i,1}$ or are an element of P', where P' is as defined in Lemma 5.7.*

Proof Let c^*, P' and P'' be as defined in Lemma 5.7. We use the enumeration of candidate edge sets from Lemma 5.6, and only keep the endpoints that are to the right of t_i. To each of these endpoint sets, we add at most c^* endpoints from P', and we add at most one endpoint from P''. There are $\text{poly}(k)$ and $O(n)$ ways to add such endpoints, respectively. This gives a family of $2^{O(k^{1-1/d})} \cdot \text{poly}(k) \cdot n = 2^{O(k^{1-1/d})}n$ sets. By Lemma 5.7, the resulting family of endpoint sets contains V. □

The rank-based approach is independent of the drums used. This brings us to the algorithm itself. It requires only a single change: all separators to the left of a previously found separator can be ignored.

For the third and final change, we can use walls to decrease the number of viable distant points. If we find a wall sufficiently far to the right of t_i, then by Lemma 5.4 all points that are sufficiently far to the right to this wall are unviable and can therefore be left out of P''. Specifically, let p_j be the leftmost point with $x_j > X(i) + \delta/2 + \delta^2$ and j divisible by 4. Note that this point is at least δ^2 to the right of $\tau_{i,0}$ and therefore at least δ^2 to the right of t_i. We then check whether $(p_j, \ldots, p_{j+3})$ is a wall. If it is not, we check $(p_{j+4}, \ldots, p_{j+7})$ and so forth, until we either have found a wall, or have reached p_n. If we find a wall, then we only use the distant points that are not rendered unviable by the wall. If we do not, then we use the same set of distant points as we used previously.

Step 4: Analysis of the running time. As stated before, the sorting of the points can be done in $O(n)$ expected time. For the rest of the analysis, we will need a few lemmas.

Observation 5.9 *Let X, Y be two integer nonnegative random variables, such that for all $k \geqslant 0$, the equation $\mathbb{P}[X \leqslant k] \geqslant \mathbb{P}[Y \leqslant k]$ holds. Let $f(k)$ be an increasing nonnegative function such that $\mathbb{E}[f(Y)] < \infty$. Then*

$$\mathbb{E}[f(X)] \leqslant \mathbb{E}[f(Y)].$$

For the proof, see [30, Formula 3.3 on p. 6].

The next lemma gives us information about among others the running time of an algorithm when the input has a random size.

Lemma 5.10 *Let $f(x)$ be a function such that $f(x) = 2^{O(x^{1-1/d})}$ for some $d \geqslant 2$. Let X be a Poisson distributed random variable with mean z. Then $\mathbb{E}[f(X)] = 2^{O(z^{1-1/d})}$.*

Proof Let $M, \lambda > 0$ be such that for all $x > 0$ we have $f(x) \leqslant M + 2^{\lambda x^{1-1/d}}$. Let us write $\mu := e^{\lambda+2}$. We get

$$\begin{aligned}
\mathbb{E}[f(X)] - M &\leqslant \mathbb{E}[2^{\lambda X^{1-1/d}}] \\
&\leqslant 2^{\lambda \lfloor \mu z \rfloor^{1-1/d}} \mathbb{P}[X \leqslant \lfloor \mu z \rfloor] + \sum_{k=\lceil \mu z \rceil}^{\infty} 2^{\lambda k^{1-1/d}} \mathbb{P}[X = k] \\
&= 2^{\lambda \lfloor \mu z \rfloor^{1-1/d}} \mathbb{P}[X \leqslant \lfloor \mu z \rfloor] + \sum_{k=\lceil \mu z \rceil}^{\infty} 2^{\lambda k^{1-1/d}} \frac{z^k e^{-z}}{k!} \\
&< 2^{\lambda \mu z^{1-1/d}} + \sum_{k=\lceil \mu z \rceil}^{\infty} 2^{\lambda k^{1-1/d}} \frac{z^k e^{-z}}{(k/e)^k} \\
&< 2^{\lambda \mu z^{1-1/d}} + \sum_{k=\lceil \mu z \rceil}^{\infty} e^{\lambda k} \frac{z^k e^k}{(\mu z)^k} \\
&= 2^{\lambda \mu z^{1-1/d}} + \sum_{k=\lceil \mu z \rceil}^{\infty} e^{-k} = 2^{\lambda \mu z^{1-1/d}} + O(1).
\end{aligned}$$

We have now shown that $\mathbb{E}[f(X)] \leqslant 2^{\lambda \mu z^{1-1/d}} + M + O(1)$, finishing the proof. □

This brings us to the result we need:

Lemma 5.11 *Let X be a Poisson distributed random variable with mean z. Let $i \geqslant 0$ be an arbitrary but fixed integer. Let j be any integer with $i \leqslant j$. Let M be a random variable defined by $\mathbb{P}[M = k] := \mathbb{P}[X = k]/\mathbb{P}[X \in \{i, \ldots, j\}]$ for $k \in \{i, \ldots, j\}$, and $\mathbb{P}[M = k] = 0$ otherwise. Let $f(x)$ be such that $f(x) = 2^{O(x^{1-1/d})}$. Then $\mathbb{E}[f(M)] = 2^{O(z^{1-1/d})}$.*

Proof Let $\lambda, \mu > 0$ be such that $g(x) := \mu 2^{\lambda x^{1-1/d}}$ is larger than $f(x)$ for all x. Clearly, it is sufficient to show that $\mathbb{E}[g(M)] = 2^{O(z^{1-1/d})}$.

Let $X' := X + i$. First, we will show that $\mathbb{P}[M \leqslant k] \geqslant \mathbb{P}[X' \leqslant k]$ for all k:

- Suppose $0 \leqslant k \leqslant i - 1$. Since $\mathbb{P}[M \leqslant k] = \mathbb{P}[X' \leqslant k] = 0$, the statement indeed holds.
- Suppose $k \geqslant j$. Since $\mathbb{P}[M \leqslant k] = 1$ and $\mathbb{P}[X' \leqslant k] < 1$, the statement indeed holds.
- Suppose $i \leqslant k \leqslant j - 1$. For this case, let us first take a look at the value of $\mathbb{P}[M = m] - \mathbb{P}[X' = m]$:

$$\begin{aligned}
\mathbb{P}[M = m] - \mathbb{P}[X' = m] &= \frac{e^{-z} z^m}{m! \mathbb{P}[i \leqslant X \leqslant j]} - \frac{e^{-z} z^{m-i}}{(m-i)!} \\
&= \frac{e^{-z} z^{m-i}}{m!} \left(\frac{z^i}{\mathbb{P}[i \leqslant X \leqslant j]} - \frac{m!}{(m-i)!} \right).
\end{aligned}$$

Note that $\mathbb{P}[M = m] - \mathbb{P}[X' = m] < 0$ if and only if m is large enough. Therefore, $\mathbb{P}[M \leqslant k] - \mathbb{P}[X' \leqslant k]$ is minimal at $k = i - 1$ or $k = j$. We have already seen that for both of these values of k, we have $\mathbb{P}[M \leqslant k] - \mathbb{P}[X' \leqslant k] \geqslant 0$. Therefore, for all $i \leqslant k \leqslant j - 1$, we have $\mathbb{P}[M \leqslant k] \geqslant \mathbb{P}[X' \leqslant k]$.

Since $\mathbb{P}[M \leqslant k] \geqslant \mathbb{P}[X' \leqslant k]$ for all k, and g is increasing, Observation 5.9 gives us that $\mathbb{E}[g(M)] \leqslant \mathbb{E}[g(X')]$. Furthermore, since

$$g(x + i) = \mu 2^{\lambda(x+i)^{1-1/d}} \leqslant \mu 2^{\lambda(x^{1-1/d}+i)} = 2^{\lambda i} g(x),$$

we get

$$\mathbb{E}[g(X')] = \mathbb{E}[g(X + i)] \leqslant 2^{\lambda i}\mathbb{E}[g(X)] = O(\mathbb{E}[g(X)]).$$

Since X follows a Poisson distribution with mean z, Lemma 5.10 gives us that $\mathbb{E}[g(X)] = 2^{O(z^{1-1/d})}$. In conclusion, we have

$$\mathbb{E}[f(M)] \leqslant \mathbb{E}[g(M)] \leqslant \mathbb{E}[g(X')] = O(\mathbb{E}[g(X)]) = 2^{O(z^{1-1/d})},$$

as we wanted to prove. □

We are now ready to analyse the total running time of the algorithm. We will look at each part of the algorithm separately.

- **Line** 1: **Computing the separators.** There are $O(n)$ separators to be computed. Recall that $\tau_{i,j} = [(X(i) - 1/2 + j)\delta, (X(i) + 1/2 + j)\delta] \times \text{Ball}^{d-1}(\delta/2)$, with $X(i) := (i + 1/2)x_n/n$ for $j \in \{-1, 0, 1\}$. By Lemma 5.6, each separator is computed in $O(M^{d+1})$ time, where M is the number of points in $\tau_{i,-1} \cup \tau_{i,0} \cup \tau_{i,1}$. Now we want a bound on the total expected amount of time needed per separator, $\sum_{k=0}^{\infty} k^{d+1}\mathbb{P}[M = k]$. Recall that $\Delta_j \sim \text{Exp}(1)$ for all j, and the total horizontal interval covered by $\tau_{i,-1} \cup \tau_{i,0} \cup \tau_{i,1}$ is 3δ. Clearly, $M \geqslant 0$. Therefore, M is almost a Poisson distributed variable with expected value 3δ. There is one difference: since there are only n points, $M \leqslant n$. Therefore, M does have all the required properties for Lemma 5.11, so the total expected time needed per separator is $2^{O((3\delta)^{1-1/d})} = 2^{O(\delta^{1-1/d})}$. (Using [26], we can prove that the expected time is $O(\delta^{d+1})$, but the above will suffice.) Since there are $O(n)$ separators, the total expected time needed for this part is $2^{O(\delta^{1-1/d})}n$.
- **Lines** 2 **and** 5: **The sets** $\mathcal{B}_i$. We will now give an upper bound on the expected value of $\sum_{i=1}^{|\mathcal{T}|+1} |\mathcal{B}_i|$. Recall that each $B \in \mathcal{B}_i$ consists of (i) the rightmost endpoints of a candidate edge set, (ii) at most c^* points from P' and (iii) at most one point to the right of P', not rendered unviable by a wall. Note that the expected number of possibilities of each type is dependent: the more possibilities one set contains, the fewer others can have. After all, no point can be in two sets of different types at once. Aside from the number of points in each interval, the three types are otherwise independent. Therefore, to find an upper bound on $\mathbb{E}[|\mathcal{B}_i|]$ it suffices to multiply upper bounds of the expected values of possibilities of (i), (ii) and (iii).

Let us start with the number of viable distant points, (iii). Let p_j be the leftmost point such that $(p_j, \ldots, p_{j+3})$ is the first candidate wall checked. We have already seen that the probability that it is a wall is some constant α. If it is not a wall, the next set we check has the same probability, and so forth. Therefore, the geometric distribution with probability α forms an upper bound on the expected number of possible walls checked. The expected x-coordinate of the last viable distant point therefore is $X(i) + O(\delta^2) + O(1/\alpha) + O(\delta^2) = x(t_i) + O(\delta^2)$. Therefore, the expected number of viable distant points is $O(\delta^2)$.

This brings us to the number of possible B'. Since we use Corollary 5.8, this is $\delta^2 \cdot 2^{O(M^{1-1/d})}$, where M is the number of points which are in $\tau_{i,-1} \cup \tau_{i,0} \cup \tau_{i,1}$ or in P'. Note that we have a factor $O(\delta^2)$ instead of the factor $O(n)$ as in the statement of Corollary 5.8. This is because, as we have just shown, the expected number of distant points we need to consider is not $O(n)$ but $O(\delta^2)$. Now, let us look at M. Recall how P' was defined: Let z be the leftmost point with an x-coordinate larger than $(X(i)+3)\delta$. Let p_j be the leftmost point more than $3\delta/2$ to the right of z. Then P' contains all points between $(X(i) + 3/2)\delta$ (the right border of $\tau_{i,1}$) and p_j. Therefore, M equals the number of points in an interval of length $3\delta + 3\delta/2 + 3\delta/2 = 6\delta$, plus one. We once more apply Lemma 5.11, giving us that the total expected number of B' is $2^{O(\delta^{1-1/d})}$.

Since $i \in \{1, \ldots, |\mathcal{T}| + 1\} \subseteq \{1, \ldots, n\}$, the total expected number of elements in all $\mathcal{B}_i$ combined is bounded by $\delta^2 \cdot 2^{O(\delta^{1-1/d})} n = 2^{O(\delta^{1-1/d})} n$.

– **Lines** 7–16. The running time of *TSP-repr* is $T(|P|, |B|) = 2^{O(|P|^{1-1/d} + |B|)}$ [12, Lem. 8]. By Lemma 5.6 we have $|B| = O(M_1^{1-1/d})$, so the running time of each call to *TSP-repr* in Algorithm 1 is $2^{O(M_2^{1-1/d})}$, for some M_1, M_2 with all properties required for Lemma 5.11. Similar statements hold for the number of weighted matchings in the representative set returned by *TSP-repr* and the number of matchings in the representative set of $A[i-1, B']$. Consequently, before executing the reduction in Line 16, the set $A[i, B]$ contains at most $2^{O(M_1^{1-1/d})} \cdot 2^{O(M_2^{1-1/d})} = 2^{O(M_3^{1-1/d})}$ entries for some M_1, M_2, M_3 with all properties required for Lemma 5.11. By Lemma 4.7 the application of the *Reduce* algorithm results in a representative set of size at most $2^{|B|-1} = 2^{O(M^{1-1/d})}$ for some M with all properties required for Lemma 5.11.

In conclusion, the total expected running time is bounded by $2^{O(\delta^{1-1/d})} n$ when the Δ_i are exponentially distributed.

Finally, we will show that the requirements for Lemma 5.5 hold, where we take $\mathcal{A}$ to be the algorithm described above. The only nontrivial requirement is that $T_{\mathcal{A}}(P_\lambda) \leqslant T_{\mathcal{A}}(P)$ for all point sets P and x-axis scaling factors $\lambda > 1$. Intuitively, we would expect this to hold: every single separator t_i and wall $(p_i, \ldots, p_{i+3})$ retains its properties when scaled with a factor larger than 1, while the boundary sets can only decrease in size and P_λ might contain extra walls. Furthermore, since we only check whether $(p_i, \ldots, p_{i+3})$ forms a wall for i divisible by 4, we do not miss walls we previously would have seen.

Unfortunately, $T_{\mathcal{A}}(P_\lambda) \leqslant T_{\mathcal{A}}(P)$ does not necessarily hold for all λ and P, since we have only proven bounds on the running times, not the actual running times themselves.

However, it is easy to show the existence of an algorithm $\mathcal{B}$ of which we know the exact expected running time of every step. Algorithm $\mathcal{B}$ is simply algorithm $\mathcal{A}$, but after every step it waits as long as necessary to make its expected running time for that step equal to the bound calculated for this step. To be precise, there are two types of waiting, best explained by an example. We have previously shown that computing a separator takes $O(|Q|^{d+1})$ time, where Q is the set of points of P in $\tau_{i,-1} \cup \tau_{i,0} \cup \tau_{i,1}$. Let c_1 be such that for all Q, computing this separator takes $T(Q) < c_1|Q|^{d+1}$ time for $\mathcal{A}$. When algorithm $\mathcal{B}$ computes a separator, it waits for $c_1|Q|^{d+1} - T(Q)$ time afterwards, therefore taking exactly $c_1|Q|^{d+1}$ time in total. Note that the time *waited* by $\mathcal{B}$ depends on Q, but the *total* time used depends only on $|Q|$, and not Q itself. This is the end of the first type of waiting, but not of the example. Next, we showed that the total *expected* time needed to compute a separator was upper bounded by $2^{O(\delta^{1-1/d})}$. So, let c_2 be such that $\mathcal{A}$, including the first type of waiting, takes $T(\delta, d) < 2^{c_2\delta^{1-1/d}}$ expected time to compute a separator. When computing a separator, $\mathcal{B}$ also waits $2^{c_2\delta^{1-1/d}} - T(\delta, d)$ time. Note that the time waited is independent of Q. Together, these two types of waiting ensure that (i) the time needed by $\mathcal{B}$ is monotone in $|Q|$ and (ii) the total expected time needed by $\mathcal{B}$ equals the calculated upper bound for $\mathcal{A}$. Finally, we always wait as long as needed to emulate the separators being in the worst possible place for that specific step. Therefore, the time taken is completely independent of the locations of the separators.

Since we know the exact running time of $\mathcal{B}$, using the arguments given earlier we get that $T_{\mathcal{B}}(P_\lambda) \leqslant T_{\mathcal{B}}(P)$ for all point sets P and x-axis scaling factors $\lambda > 1$. Therefore, we can apply Lemma 5.5 on $\mathcal{B}$, giving us that $\mathcal{B}$ has an expected running time of $2^{O(\delta^{1-1/d})}n$ on X_n, the random point set of which the x-coordinates are picked uniformly from $[0, n]$. Since $\mathcal{B}$ is slower than $\mathcal{A}$, we conclude that the expected running time of $\mathcal{A}$ must have the same bound, finishing the proof.

6 A More Efficient Algorithm for Sparse Point Sets

Recall that a point set P inside a δ-cylinder is *sparse* if for every $x \in \mathbb{R}$ the set $[x, x+1] \times \text{Ball}^{d-1}(\delta/2)$ contains at most $O(1)$ points. We have already proven that if P is sparse, we can solve EUCLIDEAN TSP in $2^{O(\delta^{1-1/d})}n^2$ time; see Theorem 4.1. The goal of this section is to obtain the following improved result:

Theorem 6.1 *Let P be a set of n points in a δ-cylinder. If P is sparse then we can solve* EUCLIDEAN TSP *in $2^{O(\delta^{1-1/d})}n + O(\delta^2 n^2)$ time.*

Our algorithm will be a dynamic program which uses three different tables. The first two of these will be very similar to the table used by our previous sparse point set algorithm. We will proceed in four steps. First, we will recall a few core concepts, and introduce another set of separators. Second, we will introduce the three tables, and define what values they contain. Third, we will show how to compute these values. Fourth, we will show that the algorithm runs in the desired time.

Step 1: A new set of separators. Because P is sparse, the drum $\sigma_i := [(i-1)\delta, i\delta] \times \text{Ball}^{d-1}(\delta/2)$ contains at most k points from P for any $i \in \mathbb{Z}$. Furthermore,

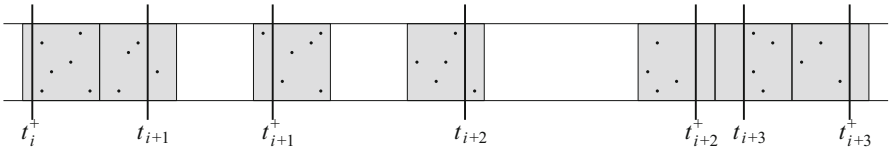


Fig. 16 The separators t_i and t_i^+ are placed alternately. Note that even though t_{i+2}^+ is placed in a different drum than t_{i+3}, in this example they are equivalent, since they induce the same partitioning of P

recall that we have a separator s_i between every pair of consecutive points p_i and p_{i+1}. These separators form the set $\mathcal{S} := \{s_1, \ldots, s_{n-1}\}$. Since any two separators that induce the same partitioning of P are considered equivalent, $\mathcal{S}$ is the set of all nontrivial separators. Recall that we also selected a separator from every second nonempty drum. Let t_i denote the separator in the i'th such drum. We define a function st that maps the index i of a separator in $\mathcal{T}$ to the index of the corresponding separator in $\mathcal{S}$. Thus $s_{\mathrm{st}(i)} = t_i$. Let $\mathcal{T} = \{t_1, \ldots, t_{|\mathcal{T}|}\}$ be the collection of selected separators. For every separator t_i, we computed a family $\mathcal{B}_i$ of candidate endpoint sets in $2^{O(k^{1-1/d})} \cdot n$ time.

We will now place separators in the other nonempty drums. Each drum contains at most k points, and has width δ. Therefore, for all i we can find a separator t_i^+ in the nonempty drum between t_i and t_{i+1} such that $x_{st^+(i)+1} - x_{st^+(i)} > \delta/(2k)$, where st^+ is defined by $s_{st^+(i)} = t_i^+$ for $1 \leqslant i \leqslant |\mathcal{T}| - 1$. See Fig. 16 for an example. Note that this is not tight; we can in fact obtain $x_{st^+(i)+1} - x_{st^+(i)} \geqslant \delta/(k+1)$, but the above will suffice. Let $\mathcal{T}^+ := \{t_1^+, \ldots, t_{|\mathcal{T}|-1}^+\}$ be the resulting set of separators. Note that no two separators t_i^+, t_j^+ are equal. However, we can have $t_i = t_i^+$ or $t_i^+ = t_{i+1}$, namely when t_i (or t_{i+1}) and t_i^+ induce the same partitioning as the boundary between the consecutive drums in which they are placed. See Fig. 16 for an example. In this case, we will still consider t_i^+ to be to the right of t_i and to the left of t_{i+1}. Recall that we also defined t_0 and $t_{|\mathcal{T}|+1}$ as 'separators' which are to the left and to the right of all points, respectively. To ensure we have no two t_i, t_{i+1} without a t_i^+ in between, we define two new separators. We define $t_0^+ := t_0$, but we do consider it to be to the right of t_0. Similarly, $t_{|\mathcal{T}|}^+ := t_{|\mathcal{T}|+1}$, which we consider to be to the left of $t_{|\mathcal{T}|+1}$. The following observation, which is a slightly modified version of Observation 5.2, shows how we can use the separators from $\mathcal{T}^+$.

Observation 6.2 *Let k be such that P contains at most k points in any drum. Let $t_i^+ \in \mathcal{T}^+$. Let r_1, r_2 be two points with $x(r_1) + 2k\delta < x_{st^+(i)}$ and $x_{st^+(i)+1} + 2k\delta < x(r_2)$. Let T be an optimal tour on P, and suppose that T contains $r_1 r_2$. Then T is bitonic at t_i^+.*

The proof is analogous to the proof of Observation 5.2; see Appendix B for the full proof. Finally, we will need the following observation:

Observation 6.3 *Let k be such that P contains at most k points in any drum. Let p_i and p_j be two points with $i < j$. Suppose $j - i \geqslant (m+1)k$ for some $m \in \mathbb{N}$. Then $x_j - x_i \geqslant m\delta$.*

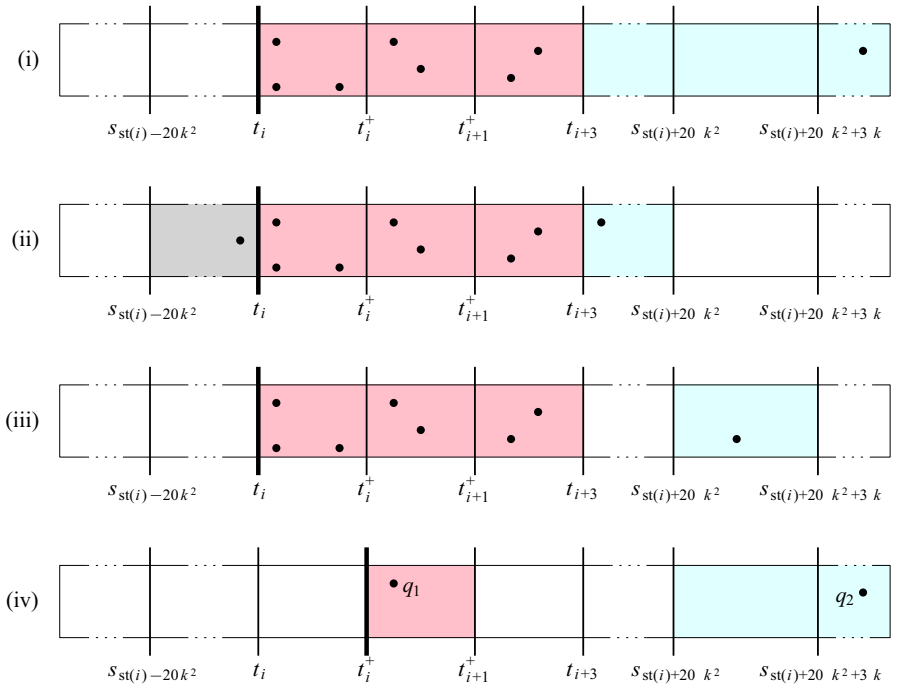


Fig. 17 Examples of entries of every table. Note that all three different types of separators are used (s, t and t^+). Also note that the spaces between consecutive separators contain varying numbers of points. (i) For comparison, an entry of table A of the basic algorithm. A set of points in $\mathcal{B}_i$ consists of a set from $\mathcal{R}_1$ (in red), and possibly any point to the right of t_{i+3} (in blue). (ii) An entry of $A^{(1)}$. A set of points in $\mathcal{B}_i^{(1)}$ contains points from a set in $\mathcal{R}_1$ (in red), possibly a point between t_{i+3} and $s_{\mathrm{st}(i)+15k^2}$ (in blue), and possibly a point between $s_{\mathrm{st}(i)-15k^2}$ and t_j (in grey). Not shown is the property that there exists a j such that all the points lie in $P[j, j+20k^2]$. (iii) An entry of $A^{(2)}$. A set of points in $\mathcal{B}_i^{(2)}$ contains points from a set in $\mathcal{R}_1$ (in red), and exactly one point between $s_{\mathrm{st}(i)+20k^2}$ and $s_{\mathrm{st}(i)+20k^2+3k}$ (in blue). (iv) An entry of $A^{(3)}$. It contains exactly one point q_1 between t_i^+ and t_{i+1}^+ (in red), and exactly one point q_2 to the right of $s_{\mathrm{st}(i)+20k^2}$ (in blue). Not shown is the property that the neighbour of q_2 is a point in $P[1, \mathrm{st}(i) - 5k^2]$

Proof Since for all $i \in \mathbb{Z}$ the drum σ_i contains at most k points, $x_j - x_i \geqslant \left\lfloor \frac{j-i}{k} - 1 \right\rfloor \delta$. From this, the observation directly follows. □

Step 2: The three tables. Let t_i be a separator in $\mathcal{T}$. Recall that the basic algorithm used a family $\mathcal{B}_i$ of sets, such that $\mathcal{B}_i$ contains the endpoint configuration of the edges of an optimal tour that are crossing t_i. The set $\mathcal{B}_i$ was formed as follows (see also Corollary 4.6). Let $\mathcal{R}_{1a}$ be the family of sets of points formed by taking the right endpoints of edges of the candidate edge sets from Lemma 4.3. Let $\mathcal{R}_{1b}$ be the family of sets of at most c^* points between t_i and t_{i+3}, where c^* is as defined in Lemma 4.5. To improve readability, we write $\mathcal{R}_1 := \{R_{1a} \cup R_{1b} | R_{1a} \in \mathcal{R}_{1a} \text{ and } R_{1b} \in \mathcal{R}_{1b}\}$. Finally, let $\mathcal{R}_2$ be the family of sets containing at most one point to the right of t_{i+3} and none to its left, so each element of $\mathcal{R}_2$ is either the empty set or a set containing

a single point. Then

$$\mathcal{B}_i = \{R_1 \cup R_2 \mid R_1 \in \mathcal{R}_1 \text{ and } R_2 \in \mathcal{R}_2\}.$$

See Fig. 17i for an example.

Recall that the basic algorithm takes $2^{O(\delta^{1-1/d})}n^2$ time because for each of the $O(n)$ separators t_i, the family $\mathcal{B}_i$ contains $2^{O(k^{1-1/d})}n$ sets. To counter this, we replace $\mathcal{B}_i$ by two other families of sets of points, each containing only $2^{O(k^{1-1/d})}$ sets. Recall that $P[i, j] := \{p_i, \ldots, p_j\}$ for all $i \leqslant j$.

Let $\mathcal{R}'_2$ be the family of sets of at most one point from $P[\mathrm{st}(i) + 1, \mathrm{st}(i) + 20k^2]$. Note that $\mathcal{R}'_2$ contains $20k^2+1$ sets: $20k^2$ singleton sets and the empty set. The number 20 is not special; it is merely a large enough constant. Finally, let $\mathcal{R}_3$ be the family of sets of at most one point from $P[\mathrm{st}(i) - 20k^2 + 1, \mathrm{st}(i)]$.

Now we define $\mathcal{B}_i^{(1)}$ as the family of sets of points formed by combining one set from $\mathcal{R}_1$, one from $\mathcal{R}'_2$ and one from $\mathcal{R}_3$, such that there exists a j such that all the points lie in $P[j, j + 20k^2]$. In other words,

$$\begin{aligned}\mathcal{B}_i^{(1)} := \{B = R_1 \cup R'_2 \cup R_3 \mid R_1 \in \mathcal{R}_1, R'_2 \in \mathcal{R}'_2, R_3 \in \mathcal{R}_3 \\ \text{and } B \subseteq P[j, j + 20k^2] \text{ for some } j\}.\end{aligned}$$

See Fig. 17ii for an example. Note that the empty set is an element of $\mathcal{B}_i^{(1)}$. Also, note that not every element of $\mathcal{B}_i^{(1)}$ is an element of $\mathcal{B}_i$. Specifically, this is true for the elements which contain a point to the left of t_i.

Let $\mathcal{R}''_2$ be the family of sets of exactly one point from $P[\mathrm{st}(i) + 20k^2 + 1, \mathrm{st}(i) + 20k^2 + 3k]$. We define

$$\mathcal{B}_i^{(2)} := \{R_1 \cup R''_2 \mid R_1 \in \mathcal{R}_1 \text{ and } R''_2 \in \mathcal{R}''_2\}.$$

Note that $\mathcal{B}_i^{(2)} \subseteq \mathcal{B}_i$ and that $\mathcal{B}_i^{(1)} \cap \mathcal{B}_i^{(i)} = \emptyset$. See Fig. 17(iii) for an example.

Now we can define the tables $A^{(1)}$, $A^{(2)}$ and $A^{(3)}$ that our algorithm uses. Recall that for Algorithm 1, we used

$$A[i, B] := \begin{cases}\text{A representative set containing pairs } (M, x), \text{ where } M \text{ is a perfect} \\ \text{matching on } B \in \mathcal{B}_i \text{ and } x \text{ is a real number equal to the minimum} \\ \text{total length of a path cover of } P_0 \cup \cdots \cup P_{i-1} \cup B \text{ realizing the} \\ \text{matching } M.\end{cases}$$

We use the same definition for $A^{(1)}[i, B]$ for all $B \in \mathcal{B}_i^{(1)}$, and for $A^{(2)}[i, B]$ for all $B \in \mathcal{B}_i^{(2)}$. We get

$$A^{(1)}[i, B] := \begin{cases}\text{A representative set containing pairs } (M, x), \text{ where } M \text{ is a perfect} \\ \text{matching on } B \in \mathcal{B}_i^{(1)} \text{ and } x \text{ is a real number equal to the minimum} \\ \text{total length of a path cover of } P_0 \cup \cdots \cup P_{i-1} \cup B \text{ realizing the} \\ \text{matching } M\end{cases}$$

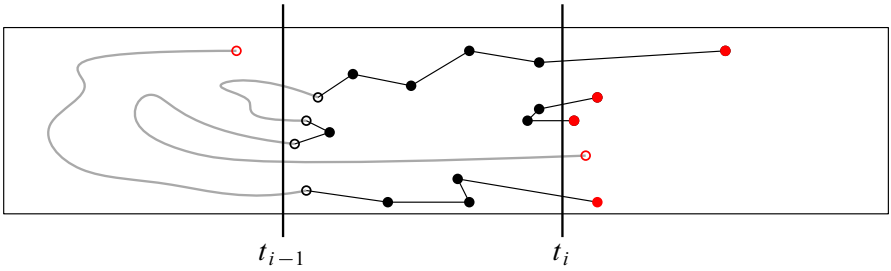


Fig. 18 An example of the computation of one of the elements of a table entry of $A^{(1)}$. The red circles (open and filled) form B. The open circles (black and red) form B'. The grey paths contain only points to the left of t_{i-1}

and

$$A^{(2)}[i, B] := \begin{cases} \text{A representative set containing pairs } (M, x), \text{ where } M \text{ is a perfect} \\ \text{matching on } B \in \mathcal{B}_i^{(2)} \text{ and } x \text{ is a real number equal to the minimum} \\ \text{total length of a path cover of } P_0 \cup \cdots \cup P_{i-1} \cup B \text{ realizing the} \\ \text{matching } M. \end{cases}$$

Note that the only difference between these three definitions is for which B the table entries are defined. Furthermore, note that $A^{(1)}[|\mathcal{T}| + 1, \emptyset]$ is defined, and the corresponding representative set contains a single element $(\emptyset, x)$ where x is the length of the shortest tour on all points of P.

Unfortunately, we cannot compute every value of $A^{(1)}$ and $A^{(2)}$ using only these two tables. This brings us to the third and final table. Let $t_i^+ \in \mathcal{T}^+$, and let q_1 be a point between t_i^+ and t_{i+1}^+. Let q_2 be a point in $P[\text{st}(i) + 20k^2 + 1, n]$. We then define

$$A^{(3)}[i, q_1, q_2] := \begin{cases} \text{The length of the shortest path from } q_1 \text{ to } q_2 \text{ that visits all points} \\ \text{in } P[1, \text{st}(i)], \text{ such that the neighbour of } q_2 \text{ is a} \\ \text{point in } P[1, \text{st}(i) - 5k^2]. \end{cases}$$

See Fig. 17(iv) for an example.

Step 3: Computing the table values. Since the algorithm uses three different tables, the order in which to compute these is nontrivial. Furthermore, for the third table, we will need to precompute two (significantly smaller) tables. Algorithm 2 (which can be found below) gives our algorithm in pseudocode, which we will explain below.

Step 3-a: Computing table values of $A^{(1)}$. Computing the values of $A^{(1)}$ is analogous to computing the table values of the basic algorithm. See Fig. 18 for an example. However, there is one exception. Sometimes, this would require a (non-existing) table entry $A^{(1)}[i - 1, B']$ for some $B' \notin \mathcal{B}_{i-1}^{(1)}$. Note that this can only be the case because B' contains a point too far to the right. Since there are at most $3k$ points between t_i and t_{i-1}, the set B' contains a point in $P[\text{st}(i-1) + 20k^2 + 1, \text{st}(i-1) + 20k^2 + 3k]$. Since there exists a j such that $B' \subseteq P[j, j + 20k^2]$, the set B' does not contain a point to the left of t_{i-1}. Therefore, $B' \in \mathcal{B}_{i-1}^{(2)}$, and so $A^{(2)}[i - 1, B']$ is the value we need.

Algorithm 2 NarrowRectTSP-DP-v2(P, δ)

Input: A sparse point set P in $(-\infty, \infty) \times \text{Ball}^{d-1}$
Output: The length of the shortest tour visiting all points in P

Compute the separators $t_1, \ldots, t_{|\mathcal{T}|}$ using Lemma 4.3, as explained earlier.
Compute the separators $t_1^+, \ldots, t_{|\mathcal{T}|-1}^+$ as explained in Step 1.
Compute the sets $\mathcal{B}_1^{(1)}, \ldots, \mathcal{B}_{|\mathcal{T}|+1}^{(1)}$ as explained in Step 2.
Compute the sets $\mathcal{B}_1^{(2)}, \ldots, \mathcal{B}_{|\mathcal{T}|+1}^{(2)}$ as explained in Step 2.
$A^{(1)}[0, \emptyset] := \{(\emptyset, 0)\}$
for $i = 1$ to $|\mathcal{T}| + 1$ **do**
  for all $B \in \mathcal{B}_i^{(1)}$ **do**
    Compute $A^{(1)}[i, B]$ as explained in Step 3-a.
  end for
  for all $B \in \mathcal{B}_i^{(2)}$ **do**
    Compute $A^{(2)}[i, B]$ as explained in Step 3-b.
  end for
  if $1 < i < |\mathcal{T}|$ **then**
    for all q_1 between t_i^+ and t_{i+1}^+ **do**
      for all q' between t_{i-1}^+ and t_i^+ **do**
        Compute $D[i, q_1, q']$, as explained in Step 3-c.
      end for
      for all $p \in P[st^+(i) - 5k^2 - 5k + 1, st^+(i) - 5k^2]$ **do**
        Compute $E[i, p, q_1]$, as explained in Step 3-c.
      end for
      for all $q_2 \in P[\text{st}(i) + 20k^2 + 1, n]$ **do**
        Compute $A^{(3)}[i, q_1, q_2]$ as explained in Step 3-c.
      end for
    end for
  end if
end for
return length$(A^{(1)}[|\mathcal{T}| + 1, \emptyset])$

For pseudocode, for a given i and $B \in \mathcal{B}_i^{(1)}$, see Algorithm 3 below. The analysis of the running time for table $A^{(1)}$ is analogous to the analysis of the basic algorithm. Therefore, the time needed to calculate a single table entry is $2^{O(k^{1-1/d})}$.

Step 3-b: Computing table values of $A^{(2)}$. Suppose we want to compute a table entry $A^{(2)}[i, B]$. Let p_j be the rightmost point of B. Note that $p_j \in P[\text{st}(i) + 20k^2 + 1, \text{st}(i) + 20k^2 + 3k]$. Now, by Observation 6.3, since

$$j - (st^+(i-1) + 1) \geqslant \text{st}(i) + 20k^2 + 1 - (\text{st}(i) + 1) = 20k^2 \geqslant (2k+1)k,$$

we get that $x_j - x_{st^+(i-1)+1} > 2k\delta$. Therefore, if p_j has a neighbour far enough to the left, then Observation 6.2 gives us that the tour is bitonic at t_{i-1}^+.

Let M be a perfect matching on B with corresponding path cover C. Let p be the neighbour of p_j in C. Now there are two possible cases: either $p \in P[\text{st}(i) - 10k^2 + 1, \text{st}(i)]$, or $p \in P[1, \text{st}(i) - 10k^2]$. To compute the value of x, we will simply take the minimum of these two cases, illustrated in Fig. 19.

Algorithm 3 NarrowRectTSP-DP-v2-1(i, B)

```
A^(1)[i, B] := ∅
for all j ∈ {1, 2} do
    for all B' ∈ B_{i-1}^(j) where B' ⊆ P_{i-1} ∪ B do
        for all (M, x) ∈ TSP-repr(P_{i-1} ∪ B' ∪ B, B'△B)) do
            for all (M', x') ∈ A^(j)[i − 1, B'] do
                if M' and M are compatible then
                    Insert (Join(M, M'), x + x') into A^(1)[i, B]
                end if
            end for
        end for
    end for
end for
Reduce(A^(1)[i, B])
```

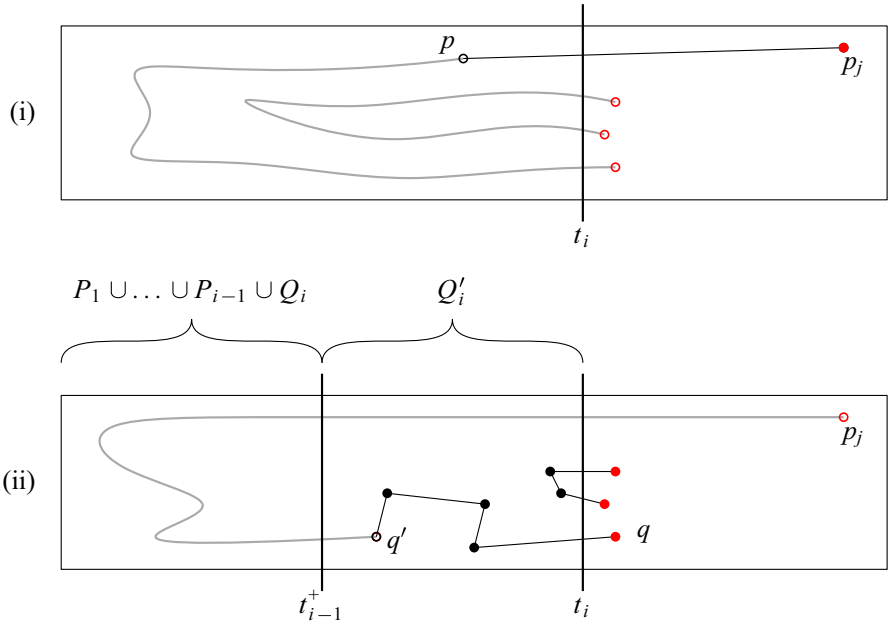


Fig. 19 The two possible cases for a table entry of $A^{(2)}$. The red circles (open and filled) form B. The open circles (black and red) form B'. The grey paths contain only points to the left of the leftmost separator shown

– **Case (i):** $p \in P[\mathrm{st}(i) - 10k^2 + 1, \mathrm{st}(i)]$. For this case, we do the following for all possible p, and take the minimum of the resulting values. We simply add the length of $|p_j p|$ to the corresponding value of $A^{(1)}[i, B']$, where B' is formed by taking B, and swapping p_j for p. See Fig. 19i for an example. We claim that $B' \in \mathcal{B}_i^{(1)}$. Since p_j, the rightmost point of B, is not in B', the only non-trivial requirement we must check for this claim is that there exists an index z such that $B' \subseteq P[z, z + 20k^2]$. We will now show that this is satisfied for $z = \mathrm{st}(i) - 10k^2$. We first take a look at the leftmost point of B', which is p. Clearly, $p \in P[\mathrm{st}(i) - 10k^2, \mathrm{st}(i) + 10k^2]$. Now, the rightmost point from B' is to the left of t_{i+3}. Since

there are at most $7k$ points between t_i and t_{i+3}, all points except p are guaranteed to be in $P[\mathrm{st}(i)+1, \mathrm{st}(i)+7k+1] \subseteq P[\mathrm{st}(i), \mathrm{st}(i)+10k^2]$. Since both the leftmost point and the rightmost point of B' are in $P[\mathrm{st}(i)-10k^2, \mathrm{st}(i)+10k^2]$, we conclude that $B' \in \mathcal{B}_i^{(1)}$.

– **Case (ii):** $p \in P[1, \mathrm{st}(i)-10k^2]$. For this case, things are slightly more complicated. By Observation 6.2, we get that C must be bitonic at t_{i-1}^+. Let M^* be a perfect matching on B. Let q be the point to which p_j is matched in M^*. See Fig. 19ii for an example. Note that the path from q to p_j of C crosses t_{i-1}^+ at least twice. Therefore, this path crosses t_{i-1}^+ exactly twice, and no other path crosses t_{i-1}^+. Let q' be the last point on this path before t_{i-1}^+ is crossed the first time. Note that $q' = q$ can hold. We can now split C into two parts of which the lengths can be computed independently: the path from q' to p_j, and the rest. Let us take a look at these two separately:

  – Recall that P_i is the set of points between t_i and t_{i+1}. We want to know the length of the shortest path from q' to p_j using all points in $P_1 \cup \ldots \cup P_{i-2} \cup Q_{i-1}$, where Q_{i-1} is the set of points between t_{i-1} and t_{i-1}^+. This is the value $A^{(3)}[i-1, q', p_j]$. Note that p_j is far enough to the right to make this a valid table entry. Indeed, since $p \in P[1, \mathrm{st}(i)-10k^2]$, and there are at most $3k$ points between t_{i-1}^+ and t_i, we have $p \in P[1, st^+(i-1)-5k^2]$, as required. Finally, we claim that q' is indeed between t_{i-1}^+ and t_i^+. We know that the tour must be bitonic at t_{i-1}^+ and t_i^+, and that there is at least one point between these two separators. If q' is not between t_{i-1}^+ and t_i^+, we would have two edges crossing both of these separators: the edge following q' in the path from q' to p and the edge from p to p_j. However, since there is at least one point between these two separators, there is also at least one point with at least one neighbour to the left of t_{i-1}^+ or to the right of t_i^+. Both lead to a contradiction, as we now have at least three edges crossing a separator where the tour must be bitonic. We conclude that q' is indeed between t_{i-1}^+ and t_i^+.
  – Let Q'_{i-1} be the set of points between t_{i-1}^+ and t_i. In other words, $Q'_{i-1} := P_{i-1} \backslash Q_{i-1}$. Let M be the matching M^*, with one difference: instead of q being matched to p_j, it is matched to q'. Now we want to know the length of the shortest path cover on $Q'_{i-1} \cup B \backslash \{p_j\}$ realizing the matching M. We can find this value using the method of the standard $2^{O(n^{1-1/d})}$ algorithm.

  We can compute the total length of C for all possible q', and take the minimum of these values, to find the minimal length of C. This ends the second case.

In conclusion, for a given i and $B \in \mathcal{B}_i^{(2)}$, we can compute $A^{(2)}[i, B]$ using the pseudocode of Algorithm 4, given on the next page. Notice that the *Reduce* at the end automatically takes the minimum of all options added at Line 6 for every M.

Next, let us take a look at the time needed to compute a single table entry of $A^{(2)}$. For the first part, there are $O(k^2)$ possible p. For every p, we need $|A^{(1)}[i, B']| = 2^{O(k^{1-1/d})}$ time. For the second part, there are $O(k)$ possible q'. For each of these combinations, we run the $2^{O(n^{1-1/d})}$ algorithm on a set containing $O(k)$ points. Analogously to the basic algorithm, the *Reduce* algorithm runs in $2^{O(k^{1-1/d})}$

Algorithm 4 NarrowRectTSP-DP-v2-2 (i, B)

$A^{(2)}[i, B] := \emptyset$
for all $p \in P[\mathrm{st}(i) - 10k^2 + 1, \mathrm{st}(i)]$ **do**
  $B' := B$ with p_j swapped for p
  for all $(M', x') \in A^{(1)}[i, B']$ **do**
    $M := M'$ with p swapped for p_j
    Insert $(M, x' + |p_j p|)$ into $A^{(2)}[i, B]$.
  end for
end for
for all $q' \in Q'_{i-1} \cup B \setminus \{p_j\}$ **do**
  $B' := \{q', p_j\}$
  $M' := \{\{q', p_j\}\}$
  $x' := A^{(3)}[i-1, q', p_j]$
  for all $(M, x) \in$ *TSP-repr*$(Q'_{i-1} \cup B' \cup B,\ B' \triangle B))$ **do**
    if M' and M are compatible **then**
      Insert (Join(M, M'), $x + x'$) into $A^{(2)}[i, B]$
    end if
  end for
end for
Reduce$(A^{(2)}[i, B])$

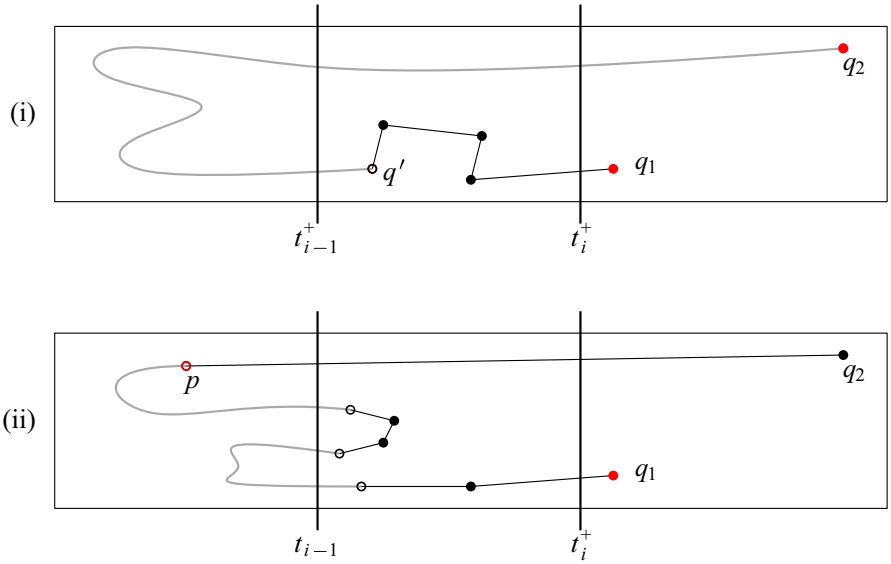


Fig. 20 The two options for a table entry of $A^{(3)}$. The red circles (open and filled) form B. The open circles (black and red) form B'. The grey paths contain only points to the left of t_{i-1}

time. Therefore, the total running time needed to compute a single table entry is $k^2 2^{O(k^{1-1/d})} + k2^{O(k^{1-1/d})} + 2^{O(k^{1-1/d})} = 2^{O(k^{1-1/d})}$.

Step 3-c: Computing table values of $A^{(3)}$. Finally, we will show how to compute the value of $A^{(3)}[i, q_1, q_2]$. The point q_2 is far enough to the right of t^+_{i-1} to make Observation 6.2 relevant. Therefore, we will once more split into two possible cases, of which we will use the minimum of the resulting values. Let p be the neighbour of

q_2 in the shortest path corresponding to this table entry. Now either $p \in P[1, st^+(i-1) - 5k^2]$, or $p \in P[st^+(i-1) - 5k^2 + 1, \mathrm{st}(i) - 5k^2]$.

- **Case (i):** $p \in P[1, st^+(i-1) - 5k^2]$. From Observation 6.3 we get that $x_{st^+(i-1)} - x(p) > 2k\delta$. Therefore, by Observation 6.2, the path from q_1 to q_2 must be bitonic at t^+_{i-1}. Since the edge pq_2 crosses t^+_{i-1}, it can only be crossed once before. See Fig. 20i for an example. Let q' be the last point on this path before t^+_{i-1} is crossed the first time. Note that $q' \neq q_1$, as q' is between t^+_{i-1} and t^+_i, since there is at least one point between these separators.
  We can now split the value of $A^{(3)}[i, q_1, q_2]$ into the sum of two values which can be computed independently: the length of the path from q_1 to q', and the length of the path from q' to q_2. Let us take a look at these two separately:
  - We want to know $D[i, q_1, q'] :=$ the length of the shortest path from q_1 to q' using all points between t^+_{i-1} and t^+_i. We can find this value using the standard $2^{O(n^{1-1/d})}$ algorithm. Note that this value is independent of q_2, and therefore only needs to be computed once. This is done at Line 16 of Algorithm 2.
  - We want to know the length of the shortest path from q' to q_2 using all points in $P_1 \cup \ldots \cup P_{i-1} \cup Q_{i-1} \cup \{q', q_2\}$. This value can be found in the table entry $A^{(3)}[i-1, q', q_2]$. Notice that this is indeed a valid table entry, since q' is between t^+_{i-1} and t^+_i.

  We can compute the sum of these lengths for all possible q', and take the minimum of these sums to find the optimal path length for this case.
- **Case (ii):** $p \in P[st^+(i-1) - 5k^2 + 1, \mathrm{st}(i) - 5k^2]$. In this case, $A^{(3)}[i, q_1, q_2]$ is equal to $|q_2 p|$ plus the length of the shortest path from p to q_1, using every point in $P[1, \mathrm{st}(i)]$. See Fig. 20ii for an example. Let $B = \{p, q_1\}$. We claim that table $A^{(1)}$ contains a representative set containing pairs (M, x), where M is a perfect matching on B and x is a real number equal to the total length of the path cover of $P[1, \mathrm{st}(i)] \cup B$ realizing the matching M. Note that since $B \subseteq P[st^+(i-1) - 10k^2 + 1, st^+(i-1) + 3k]$, no invalid table entries of $A^{(1)}$ are called. Since B contains only two points, this representative set contains only one pair, of which $M = \{\{p, q_1\}\}$. Therefore, the value x is the length of the shortest path we are looking for. Since this value is independent of q_2, it needs only to be computed once. This can be done with the following pseudocode, which is the full version of Line 19 of Algorithm 2:

```
E[i, p, q_1] := ∞
B := {p, q_1}
for all B' ∈ B_i^(1) where B' ⊆ Q_i ∪ B do
    for all (M, x) ∈ TSP-repr(Q_i ∪ B' ∪ B, B'△B)) do
        for all (M', x') ∈ A^(1)[i, B'] do
            if M' and M are compatible then
                E[i, p, q_1] := min{E[i, p, q_1], x + x'}
            end if
        end for
    end for
end for
```

This ends the second case.

Combining the two cases, for a given i, q_1 and q_2 we get the following pseudocode to use at Line 22 of Algorithm 2:

```
A^(3)[i, q1, q2] := ∞
for all q' between t_{i-1}^+ and t_i^+ do
    A^(3)[i, q1, q2] := min{A^(3)[i, q1, q2], D[i, q1, q'] + A^(3)[i − 1, q', q2]}
end for
for all p ∈ P[st^+(i − 1) − 5k^2 + 1, st(i)] do
    A^(3)[i, q1, q2] := min{A^(3)[i, q1, q2], E[i, p, q1] + |pq2|}
end for
```

Finally, let us take a look at the time needed to compute a single table entry of $A^{(3)}$. For the first option, there are $O(k)$ possible q', and the computations now only take constant time per q'. For the second option, there are $O(k)$ possible p (since $\mathrm{st}(i) - st^+(i-1) \leqslant 3k$), and the computations only take constant time per p. This brings the total time needed to compute a single table entry of $A^{(3)}$ to $O(k+k) = O(k)$. This does not include the computations done at Lines 19 and 22 of Algorithm 2; these will be considered below.

This concludes the computation of the table entries.

Step 4: The running time. Let us analyse the total time needed by Algorithm 2.

- **Computation of table entries of $A^{(1)}$ (Line 8).** There are $O(n)$ possible i and $2^{O(k^{1-1/d})}$ possible $B \in \mathcal{B}_i^{(1)}$. As seen before, the computation of a single table entry takes $2^{O(k^{1-1/d})}$ time. This brings the total time needed for this line to $n2^{O(k^{1-1/d})}2^{O(k^{1-1/d})} = 2^{O(k^{1-1/d})}n$.
- **Computation of table entries of $A^{(2)}$ (Line 11).** There are $O(n)$ possible i and $2^{O(k^{1-1/d})}$ possible $B \in \mathcal{B}_i^{(2)}$. As seen before, the computation of a single table entry takes $2^{O(k^{1-1/d})}$ time. This brings the total time needed for this line to $n2^{O(k^{1-1/d})}2^{O(k^{1-1/d})} = 2^{O(k^{1-1/d})}n$.
- **Precomputation of table D (Line 16).** We compute the length of the shortest path from q_1 to q' using all points between t_{i-1}^+ and t_i^+. There are $O(n)$ possible i, there are $O(k)$ possible q_1 and there are $O(k)$ possible q'. The $2^{O(n^{1-1/d})}$ algorithm is run on a point set of size $O(k)$. Therefore, the total time needed for this line is $2^{O(k^{1-1/d})}n$.
- **Precomputation of table E (Line 19).** Let i, p, q_1 be some valid combination for the second option. There are $O(n)$ possible i, there are $O(k)$ possible p and $O(k)$ possible q_1. Analogously to the computation of a table entry of $A^{(1)}$, for each combination $2^{O(k^{1-1/d})}$ time is needed. Therefore, this line takes $2^{O(k^{1-1/d})}n$ time in total.
- **Computation of table entries of $A^{(3)}$ (Line 22).** There are $O(n)$ possible i, a total of $O(k)$ possible q_1 and $O(n)$ possible q_2. As seen before, the computation of a single table entry takes $O(k)$ time. This brings the total time needed for this line to $O(nknk) = O(k^2n^2)$.

This brings the total running time to $2^{O(k^{1-1/d})}n + 2^{O(k^{1-1/d})}n + 2^{O(k^{1-1/d})}n + 2^{O(k^{1-1/d})}n + O(k^2n^2) = 2^{O(k^{1-1/d})}n + O(k^2n^2)$. Since $k = O(\delta)$, this concludes the proof of Theorem 6.1.

7 Concluding Remarks

Our paper contains three main results on EUCLIDEAN TSP.

First, we proved that for points with distinct integer x-coordinates in a strip of width δ, an optimal bitonic tour is optimal overall when $\delta \leqslant 2\sqrt{2}$. The proof of this bound, which is tight in the worst case, is partially automated to reduce the potentially enormous number of cases to two worst-case scenarios. It would be interesting to see whether a direct proof can be given for this fundamental result. We note that the proof of Theorem 2.1 can easily be adapted to point sets of which the x-coordinates of the points need not be integer, as long as the difference between x-coordinates of any two consecutive points is at least 1.

Second, we gave a $2^{O(\delta^{1-1/d})}n + O(\delta^2n^2)$ dynamic programming algorithm for sparse point sets in a d-dimensional δ-cylinder. For $\delta = \Theta(n)$ the running time becomes $2^{O(n^{1-1/d})}$, which is optimal under ETH. We expect that the factor $O(\delta^2n^2)$ can be improved to a factor $O(\delta^2\, n \log^2 n)$, using a method similar to that used by De Berg et al. to solve BITONIC TSP in $O(n \log^2 n)$ time [11].

Third, we gave a $2^{O(\delta^{1-1/d})}n$ expected time algorithm for random point sets. The proof also gives a way to relate the expected running times of algorithms for any problem on two different kinds of random point sets: a version where the x-coordinates of the points are taken uniformly at random from $[0, n]$, and a version where the differences between two consecutive x-coordinates are taken independently from Exp(1). Note that the problem is scalable, so these results generalize to any desired interval or desired expected difference between two consecutive x-coordinates.

There are several directions for further research. In $\mathbb{R}^2$, our results show how the complexity of TSP scales as we go from an essentially 1-dimensional problem (δ is a small constant) to an essentially unrestricted 2-dimensional problem ($\delta = n$). We can also ask how the complexity of TSP scales from a 2-dimensional problem to a 3-dimensional problem. For instance, we can assume the points are randomly distributed inside a box of size $n \times n \times \delta$, or we can consider sparse point sets inside a slab of width δ. (Here a point set is sparse when every box of size $1 \times 1 \times \delta$ contains $O(1)$ points.) The goal would be to obtain an algorithm with running time $2^{O(f(\delta)\sqrt{n})}$, where $f(n) = O(n^{1/6})$. Such a running time becomes $2^{O(\sqrt{n})}$ for constant δ (which is optimal for TSP in $\mathbb{R}^2$, under ETH), and it becomes $2^{O(n^{2/3})}$ for $\delta = n$ (which is optimal for TSP in $\mathbb{R}^3$, assuming ETH). One can also generalize the problem in another way. Instead of assuming that the points are all close to a line or line segment, one could also assume that the points are all close to for example a curve, or a (connected) set of line segments. We believe that our algorithm can serve as the basis of an algorithm solving such a problem, under the assumption that the point sets are dense enough to ensure that the solution will generally follow these curves / segments. Making this precise,

and investigating how the running time depends on the number of line segments, would be interesting.

More generally, we believe this more fine-grained look at the dimension of a problem instance is worth investigating for many other problems as well. We already investigated this for the MINIMUM RECTILINEAR STEINER TREE problem in $\mathbb{R}^2$, where the fastest known algorithm runs in $2^{O(\sqrt{n}\log n)}$ time [17]. We showed that there exists an $n^{O(\sqrt{\delta})}$ algorithm for sparse point sets and a $2^{O(\delta\sqrt{\delta})}n$ expected time algorithm for random point sets [1].

Funding This study was supported by Dutch Research council (NWO) under project no. NETWORKS-024.002.003.

Data Availability Data sharing not applicable to this article as no datasets were generated or analysed during the current study.

Appendix A: The Automated Prover

Here, we will give an overview of the inner workings of the automated prover. See Algorithm 5 for pseudocode, which we will now explain. First, all candidate sets of edges $\overline{F}'$ that form a tour together with $\widetilde{E}$ are generated. Then, a set of cases is generated. Each case consists of the following:

- The x-coordinates of each of the points, stored in $\mathcal{X}$. Every point must have a different integer x-coordinate, and must adhere to the given bounds on the x-coordinates.
- For every point, an interval denoting the range in which its y-coordinate must lie, stored in $\mathcal{Y}$. When the cases are first generated, this is simply $[0, \delta]$ for every point.

For each possible combination $\mathcal{X}$ of x-coordinates of the points, one such case is generated. Note that every possible set of points in represented by one of these cases. Then, for every case, it does the following:

- To prove a case, it calculates an upper bound on the total lengths of every candidate $\overline{F}'$, and a lower bound of the total length of $\overline{F}$. For every edge, a lower bound can be found by simply taking the points as close together as possible, and an upper bound is found by doing the opposite. See Fig. 21 for some examples. Suppose there exists a set $\overline{F}'$ such that the upper bound on its length is guaranteed to be lower than the lower bound on the length of $\overline{F}$. Then $\|\overline{F}'\| < \|\overline{F}\|$ must hold, and the case holds. To counter rounding errors, the calculated upper bound must be more than some fixed $\eta := 10^{-6}$ lower than the calculated lower bound for this to trigger. The prover has been implemented in Java using `doubles`, which have a precision of up to 15 to 16 decimal digits.
- If it fails to prove the case, it splits the case into $2^{n_{\text{left}}+n_{\text{right}}}$ cases, by splitting the interval of every y-coordinate into two equal parts. For example, if a case has two points and intervals $[0, 2]$ and $[6, 8]$, it is split into the four cases $([0, 1], [6, 7])$, $([0, 1], [7, 8])$, $([1, 2], [6, 7])$ and $([1, 2], [7, 8])$. Then, it recursively tries to prove all of these cases. If, however, the intervals become too small (smaller than the given precision parameter ε), the prover gives up on proving this case.

Algorithm 5 *FindShorterTour*$(n_{\text{left}}, n_{\text{right}}, \overline{F}, \widetilde{E}, X, \delta, \varepsilon)$

Input:
- n_{left} and n_{right} denote the number of points with x-coordinate at most -1 and at least 0, respectively;
- $\overline{F}$, $\widetilde{E}$ are edge sets such that $\widetilde{E} \cup \overline{F}$ forms a tour;
- X is an array where $X[i]$ specifies the set from which the x-coordinate of the i-th point in the given scenario may be chosen;
- ε denotes the maximum 'size' of the returned cases.

Output:

A list of *scenarios* and an *outcome* for each scenario.
A scenario contains for each point q an x-coordinate $x(q)$ from the set of allowed x-coordinates for q, and a range y-range$(q) \subseteq [0, 2\sqrt{2}]$ for its y-coordinate. This y-range is an interval of length at least $\varepsilon/2$. The outcome of a scenario is either SUCCESS or FAIL. An outcome SUCCESS means that a set $\overline{F}'$ has been found with the desired properties: $\widetilde{E} \cup \overline{F}'$ is a tour, and for all possible instantiations of the scenario—that is, all choices of y-coordinates from the y-ranges in the scenario—we have $\|\overline{F}'\| < \|\overline{F}\|$. An outcome FAIL means that such an $\overline{F'}$ has not been found. It does not guarantee that such an F' does not exist for this scenario.
The list of scenarios is complete in the sense that for any instantiation of the input case there is a scenario that covers it.

```
result ← ∅
Generate all F̄′ for which Ẽ ∪ F̄′ is a tour
for Every valid combination 𝒳 of x-coordinates for the points do
    Generate case (𝒳, 𝒴), consisting of the exact x-coordinates of the points, and the bounds [0, δ] for the y-coordinate of each of the points
    TryToProveCase(𝒳, 𝒴)
end for
return result

function TryToProveCase(𝒳, 𝒴)
    if there exists an F̄′ such that F̄′ is guaranteed to be shorter than F̄ then
        Add (𝒳, 𝒴, Success) to result
    else if the bounds in 𝒴 are larger than ε then
        for every of the 2^(n_left + n_right) subcases 𝒴′ of 𝒴 do
            TryToProveCase(𝒳, 𝒴′)
        end for
    else
        Add (𝒳, 𝒴, Fail) to result
    end if
end function
```

This process continues until all cases have been either proven or given up on. Finally, the prover returns a list of all cases and whether it succeeded or failed on these cases. Note that the smaller the precision parameter ε is, the more precise the cases (and therefore the answer) will be. However, a smaller ε will also result in more cases, which increases both the running time and the number of lines of the output.

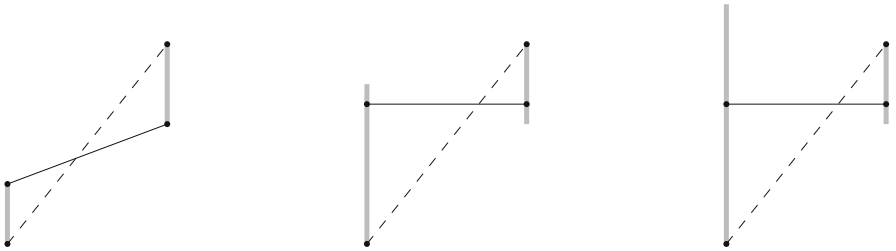


Fig. 21 Three pairs of intervals, the shortest edges between points in those intervals (solid) and the longest edges between points in those intervals (dashed)

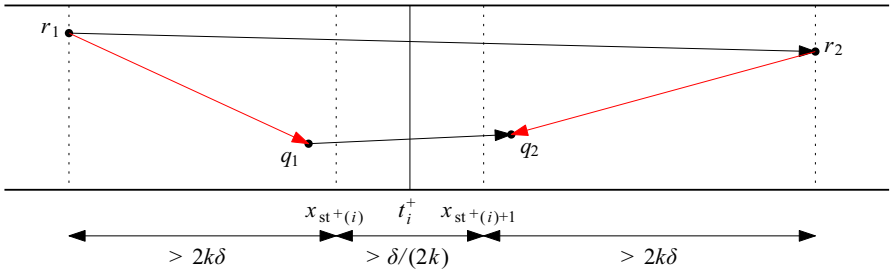


Fig. 22 An example of Observation 6.2. Edges of T are shown in black, edges of T' are shown in red

Appendix B: Omitted Proofs

Proof of Observation 6.2

See Fig. 22 for an example. Note that $x_{st^+(i)}$ is the x-coordinate of the rightmost point to the left of t_i^+, and that $x_{st^+(i)+1}$ is the x-coordinate of the leftmost point to the right of t_i^+. Therefore, $x_{st^+(i)+1} - x_{st^+(i)} > \delta/(2k)$.

W.l.o.g., T contains the directed edge from r_1 to r_2. Suppose for a contradiction that T is an optimal tour and not bitonic at t_i^+. Then T must cross t_i^+ at least four times, which implies that there must be another directed edge q_1q_2 with $x(q_1) \leqslant x_{st^+(i)}$ and $x_{st^+(i)+1} \leqslant x(q_2)$.

Note that $q_1 \neq r_1$ and $q_2 \neq r_2$, since a single point cannot have two incoming or two outgoing edges. We will now show that $|r_1q_1| + |r_2q_2| < |q_1q_2| + |r_1r_2|$. By Observation 3.1, we then get an optimal tour T' shorter than T, thereby finishing the proof. First, we will simplify the problem using an argument similar to the proof of Observation 3.2. Suppose we move q_1 towards q_2 until $x(q_1) = x_{st^+(i)}$. By doing so, $|q_1r_1|$ will never decrease more than $|q_1q_2|$. Therefore, it is sufficient to prove the statement for $x(q_1) = x_{st^+(i)}$. Analogously, it is sufficient to prove the statement for $x(q_2) = x_{st^+(i)+1}$ and $x(r_1) + 2k\delta = x(q_1)$ and $x(q_2) + 2k\delta = x(r_2)$. Finally, recall that $x_{st^+(i)+1} - x_{st^+(i)} > \delta/(2k)$, so we will prove the statement for $x_{st^+(i)+1} - x_{st^+(i)} = \delta/(2k)$. Together, we now have

$$x(r_1) + 2k\delta = x(q_1) = x(q_2) - \delta/(2k) = x(r_2) - \delta/(2k) - 2k\delta.$$

We get

$$\begin{aligned}
|q_1r_1| + |q_2r_2| &\leqslant \sqrt{(x(q_1) - x(r_1))^2 + \delta^2} + \sqrt{(x(r_2) - x(q_2))^2 + \delta^2} \\
&= 2\delta\sqrt{(2k)^2 + 1} \\
&< 2\delta(2k + 1/(2k)) \\
&= \delta/(2k) + (2(2k\delta) + \delta/(2k)) \\
&= (x(q_2) - x(q_1)) + (x(r_2) - x(r_1)) \\
&\leqslant |q_1q_2| + |r_1r_2|.
\end{aligned}$$

By Observation 3.1, this gives us an optimal tour T' shorter than T, thereby finishing the proof.

References

1. Alkema, H., de Berg, M.: Rectilinear Steiner trees in narrow strips. In: Proceedings of 37th International Symposium on Computational Geometry (SoCG), pp. 9:1–9:16 (2021)
2. Arora, S.: Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. J. ACM **45**(5), 753–782 (1998)
3. Bodlaender, H.L., Cygan, M., Kratsch, S., Nederlof, J.: Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. Inf. Comput. **243**, 86–111 (2015)
4. Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report. Graduate School of Industrial Administration, Carnegie Mellon University (1976)
5. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms, 3rd edn. MIT, Cambridge (2009)
6. Cutler, M.: Efficient special case algorithms for the n-line planar traveling salesman problem. Networks **10**, 183–195 (1980)
7. Cygan, M., Kratsch, S., Nederlof, J.: Fast Hamiltonicity checking via bases of perfect matchings. J. ACM **65**(3), 12:1-12:46 (2018)
8. Daley, D., Vere-Jones, D.: An Introduction to the Theory of Point Processes: Volume II: General Theory and Structure. Probability and Its Applications. Springer, New York (2007)
9. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry: Algorithms and Applications. Springer, Berlin (2008)
10. de Berg, M., Buchin, K., Jansen, B., Woeginger, G.: Fine-grained complexity analysis of two classic TSP variants. In: Proceedings of 43rd International Colloquium on Automata, Languages, and Programming (ICALP), pp. 5:1–5:14 (2016)
11. de Berg, M., Buchin, K., Jansen, B.M.P., Woeginger, G.J.: Fine-grained complexity analysis of two classic TSP variants. ACM Trans. Algorithms **17**(1), 1–29 (2016)
12. de Berg, M., Bodlaender, H., Kisfaludi-Bak, S., Kolay, S.: An ETH-tight exact algorithm for Euclidean TSP. In: Proceedings of 59th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 450–461 (2018)
13. Deineko, V., Woeginger, G.: The convex-hull-and-k-lines traveling salesman problem. Inf. Proc. Lett. **59**(3), 295–301 (1996)
14. Deineko, V., van Dal, R., Rote, G.: The convex-hull-and-line traveling salesman problem: a solvable case. Inf. Proc. Lett. **51**, 141–148 (1994)
15. Deineko, V., Hoffmann, M., Okamoto, Y., Woeginger, G.: The traveling salesman problem with few inner points. Oper. Res. Lett. **34**(1), 106–110 (2006)
16. Edelsbrunner, H., Rote, G., Welzl, E.: Testing the necklace condition for shortest tours and optimal factors in the plane. Theor. Comput. Sci. **66**, 157–180 (1989)

17. Fomin, F., Lokshtanov, D., Kolay, S., Panolan, F., Saurabh, S.: Subexponential algorithms for rectilinear Steiner tree and arborescence problems. ACM Trans. Algorithms **16**, 1–37 (2020). https://doi.org/10.1145/3381420
18. Garey, M., Graham, R., Johnson, D.: Some NP-complete geometric problems. In: Proceedings of the 8th Annual ACM Symposium on Theory of Computing (STOC), pp. 10–22 (1976)
19. Hwang, R., Chang, R., Lee, R.: The searching over separators strategy to solve some NP-hard problems in subexponential time. Algorithmica **9**(4), 398–423 (1993)
20. Impagliazzo, R., Paturi, R.: On the complexity of k-SAT. J. Comput. Syst. Sci. **62**(2), 367–375 (2001)
21. Kann, V.: On the approximability of NP-complete optimization problems. Ph.D. thesis, Royal Institute of Technology, Stockholm (1992)
22. Mitchell, J.: 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)
23. Papadimitriou, C.: The Euclidean traveling salesman problem is NP-complete. Theor. Comput. Sci. **4**(3), 237–244 (1977)
24. Rao, S., Smith, W.D.: Approximating geometrical graphs via 'spanners' and 'banyans'. In: Proceedings of 30th ACM Symposium on Theory of Computing (STOC), pp. 540–550 (1998)
25. Reinhold, A.: Some results on minimal covertex polygons. Manuscript, City College of New York (1965)
26. Riordan, J.: Moment recurrence relations for binomial, Poisson and hypergeometric frequency distributions. Ann. Math. Stat. **8**(2), 103–111 (1937). https://doi.org/10.1214/aoms/1177732430
27. Rote, G.: The n-line traveling salesman problem. Networks **22**, 91–108 (1992)
28. Sanders, D.: On extreme circuits. Ph.D. thesis, City University of New York (1968)
29. Smith, W., Wormald, N.: Geometric separator theorems and applications. In: Proceedings 39th Annual Symposium on Foundations of Computer Science (FOCS), pp. 232–243 (1998)
30. Thorisson, H.: Coupling, Stationarity and Regeneration. Springer, New York (2000)