An ETH-tight exact algorithm for euclidean TSP

Citation for published version (APA):
de Berg, M. T., Bodlaender, H. L., Kisfaludi-Bak, S., & Kolay, S. (2018). An ETH-tight exact algorithm for euclidean TSP. *arXiv*, *2018*, [1807.06933v2]. https://doi.org/10.48550/arXiv.1807.06933

DOI:
10.48550/arXiv.1807.06933

Document status and date:
Published: 01/01/2018

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

An ETH-Tight Exact Algorithm for Euclidean TSP*

Mark de Berg[1], Hans L. Bodlaender[1,2], Sándor Kisfaludi-Bak[1], and Sudeshna Kolay[1]

[1]Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands
{M.T.d.Berg, S.Kisfaludi.Bak, S.Kolay}@tue.nl
[2]Department of Computer Science, Utrecht University, Utrecht, The Netherlands
H.L.Bodlaender@uu.nl

August 6, 2018

Abstract

We study exact algorithms for EUCLIDEAN TSP in $\mathbb{R}^d$. In the early 1990s algorithms with $n^{O(\sqrt{n})}$ running time were presented for the planar case, and some years later an algorithm with $n^{O(n^{1-1/d})}$ running time was presented for any $d \geqslant 2$. Despite significant interest in subexponential exact algorithms over the past decade, there has been no progress on EUCLIDEAN TSP, except for a lower bound stating that the problem admits no $2^{O(n^{1-1/d-\varepsilon})}$ algorithm unless ETH fails. Up to constant factors in the exponent, we settle the complexity of EUCLIDEAN TSP by giving a $2^{O(n^{1-1/d})}$ algorithm and by showing that a $2^{o(n^{1-1/d})}$ algorithm does not exist unless ETH fails.

1 Introduction

The TRAVELING SALESMAN PROBLEM, or TSP for short, is one of the most widely studied problems in all of computer science. In (the symmetric version of) the problem we are given a complete undirected graph $\mathcal{G}$ with positive edge weights, and the goal is to compute a minimum-weight cycle visiting every node exactly once. In 1972 the problem was shown to be NP-hard by Karp [15]. A brute-force algorithm for TSP runs in $O(n!)$, but the celebrated Held-Karp dynamic-programming algorithm, discovered independently by Held and Karp [11] and Bellman [2], runs in $O(2^n n^2)$ time. Despite extensive efforts and progress on special cases, it is still open if an exact algorithm for TSP exists with running time $O(\text{poly}(n)(2-\varepsilon)^n)$.

In this paper we study the Euclidean version of TSP, where the input is a set P of n points in $\mathbb{R}^d$ and the goal is to find a tour of minimum Euclidean length visiting all the points. EUCLIDEAN TSP has been studied extensively and it can be considered one of the most important geometric optimization problems. Already in the mid-1970s, EUCLIDEAN TSP was shown to be NP-hard [10, 20]. Nevertheless, its computational complexity is markedly different from that of the general TSP problem. For instance, EUCLIDEAN TSP admits efficient approximation algorithms. Indeed, the famous algorithm by Christofides [4]—which actually works for the more general METRIC TSP problem—provides a $(3/2)$-approximation in polynomial time, while no polynomial-time approximation algorithm exists for the general problem (unless P =NP). It was a long-standing open problem whether EUCLIDEAN TSP admits a PTAS. The question was answered

*This work was supported by the NETWORKS project, funded by the Netherlands Organization for Scientific Research NWO under project no. 024.002.003.

affirmatively by Arora [1] who provided a PTAS with running time $n(\log n)^{O(\sqrt{d}/\varepsilon)^{d-1}}$. Independently, Mitchell [18] designed a PTAS in $\mathbb{R}^2$. The running time was improved to $2^{(1/\varepsilon)^{O(d)}}n + (1/\varepsilon)^{O(d)} n \log n$ by Rao and Smith [22]. Hence, the computational complexity of the approximation problem has essentially been settled.

Results on exact algorithms for EUCLIDEAN TSP—these are the topic of our paper—are also quite different from those on the general problem. The best known algorithm for the general case runs, as already remarked, in exponential time, and there is no $2^{o(n)}$ algorithm under ETH due to classical reductions for HAMILTONIAN CYCLE [5, Theorem 14.6]. EUCLIDEAN TSP, on the other hand, is solvable in subexponential time. For the planar case this has been shown in the early 1990s by Kann [14] and independently by Hwang, Chang and Lee [12], who presented an algorithm with an $n^{O(\sqrt{n})}$ running time. Both algorithms use a divide-and-conquer approach that relies on finding a suitable separator. The approach taken by Hwang, Chang and Lee is based on considering a triangulation of the point set such that all segments of the tour appear in the triangulation, and then observing that the resulting planar graph has a separator of size $O(\sqrt{n})$. Such a separator can be guessed in $n^{O(\sqrt{n})}$ ways, leading to a recursive algorithm with $n^{O(\sqrt{n})}$ running time. It seems hard to extend this approach to higher dimensions. Kann obtains his separator in a more geometric way, using the fact that in an optimal tour, there cannot be too many long edges that are relatively close together—see the Packing Property we formulate in Section 2. This makes it possible to compute a separator that is crossed by $O(\sqrt{n})$ edges of an optimal tour, which can be guessed in $n^{O(\sqrt{n})}$ ways. The geometric flavor of this algorithm makes it more amenable to extensions to higher dimensions. Indeed, some years later Smith and Wormald [23] gave an algorithm for EUCLIDEAN TSP in $\mathbb{R}^d$, which is based on a similar kind of geometric separator as used by Kann. Their algorithm runs in $n^{O(n^{1-1/d})}$ time. (Here and in the sequel we consider the dimension d to be a fixed constant.)

The main question, also posed by Woeginger in his survey [24] on open problems around exact algorithms, is the following: is an exact algorithm with running time $2^{O(n^{1-1/d})}$ attainable for EUCLIDEAN TSP? Similar results have been obtained for some related problems. In particular, Deineko *et al.* [8] proved that Hamiltonian Cycle on planar graphs can be solved in $2^{O(\sqrt{n})}$ time, and Dorn *et al.* [9] proved that TSP on weighted planar graphs can be solved in $2^{O(\sqrt{n})}$ time. Marx and Sidiropoulos [17] have recently shown that EUCLIDEAN TSP does not admit an algorithm with $2^{O(n^{1-1/d-\varepsilon})}$, unless the Exponential Time Hypothesis (ETH) [13] fails. In the past twenty years the algorithms for EUCLIDEAN TSP have not been improved, however. Hence, even for the planar case the complexity of EUCLIDEAN TSP is still unknown.

Our contribution. We finally settle the complexity of EUCLIDEAN TSP, up to constant factors in the exponent: we present an algorithm for EUCLIDEAN TSP in $\mathbb{R}^d$, where $d \geqslant 2$ is a fixed constant, with running time $2^{O(n^{1-1/d})}$, and we show that no $2^{o(n^{1-1/d})}$ algorithm exists unless ETH fails.

The lower bound follows in a straightforward manner from a recent lower bound by De Berg *et al.* [7] for HAMILTONIAN CYCLE in d-dimensional induced grid graphs; our main contribution lies in the upper bound. The global approach to obtain the upper bound is similar to the approach of Kann [14] and Smith and Wormald [23]: we use a divide-and-conquer algorithm based on a geometric separator. A geometric separator for a given point set P is a simple geometric object—we use a hypercube—such that the number of points inside the separator and the number of points outside the separator are roughly balanced. As mentioned above, Kann [14] and Smith and Wormald [23], use a packing property of the edges in an optimal TSP tour to argue that a separator exists that is crossed by only $O(n^{1-1/d})$ edges from the tour. Since P defines $\binom{n}{2}$ possible edges, the set of crossing edges can be guessed in $n^{O(n^{1-1/d})}$ ways.

The first obstacle we must overcome if we want to beat this running time is therefore that the number of subproblems is already too large at the first step of the recursive algorithm. Unfortunately there is no hope of obtaining a balanced separator that is crossed by $o(n^{1-1/d})$ edges from the tour: there are point sets such that any balanced separator that has a "simple" shape (e.g., ball or hypercube) is crossed $\Omega(n^{1-1/d})$ times by an optimal tour. Thus we proceed differently: we prove that there exists a separator such that, even though it can be crossed by up to $\Theta(n^{1-1/d})$ edges from an optimal tour, the total number of candidate subsets of crossing edges we need to consider is only $2^{O(n^{1-1/d})}$. We obtain such a separator in two steps. First we prove a *distance-based separator theorem* for point sets. Intuitively, this theorem states that any point set P admits a balanced separator such that the number of points from P within a certain distance from the separator decreases rapidly as the distance decreases. In the second step we then prove that this separator σ has the required properties, namely (i) σ is crossed by $O(n^{1-1/d})$ edges in an optimal tour, and (ii) the number of candidate sets of crossing edges is $2^{O(n^{1-1/d})}$. In order to prove these properties we use the Packing Property of the edges in an optimal tour.

There is one other obstacle we need to overcome to obtain a $2^{O(n^{1-1/d})}$ algorithm: after computing a suitable separator σ and guessing a set S of crossing edges, we still need to solve many different subproblems. The reason is that the partial solutions on either side of σ need to fit together into a tour on the whole point set. Thus a partial solution on the outside of σ imposes connectivity constraints on the inside. More precisely, if B is the set of endpoints of the edges in S that lie inside σ, then the subproblem we face inside σ is as follows: compute a set of paths visiting the points inside σ such that the paths realize a given matching on B. The number of matchings on B boundary points is $|B|^{\Theta(|B|)}$, which is again too much for our purposes. Fortunately, the rank-based approach [3, 6] developed in recent years can be applied here. By applying this approach in a suitable manner, we then obtain our $2^{O(n^{1-1/d})}$ algorithm.

A word on the model of computation. In this paper we are mainly interested in the combinatorial complexity of EUCLIDEAN TSP. The algorithm we describe through Sections 2 and 3 therefore works in the real-RAM model of computation, with the capability of taking square roots. In particular, we assume that distances can be added in $O(1)$ time, so that the length of a given tour can be computed exactly in $O(n)$ time. (There is an effort to attack the problem of comparing the sums of square roots of integers on a word RAM, see [21].) In Section 4 we also consider the following "almost Euclidean" version of the problem: we are given a set $P = \{p_1, \ldots, p_n\}$ with rational coordinates, together with a distance matrix D such that $D[i, j]$ contains an approximation of $|p_i p_j|$. The property we require is that the ordering of distances is preserved: if $|p_i p_j| < |p_k p_l|$ then $D[i, j] < D[k, l]$. We show that an optimal tour in this setting satisfies the Packing Property, which implies that our algorithm can solve the almost Euclidean version of EUCLIDEAN TSP in $2^{O(n^{1-1/d})}$ time.

2 A separator theorem for TSP

In this section we show how to obtain a separator that can be used as the basis of an efficient recursive algorithm to compute an optimal TSP tour for a given point set. Intuitively, we need a separator that is crossed only few times by an optimal solution and such that the number of candidate sets of crossing edges is small. We obtain such a separator in two steps: first we construct a separator σ such that there are only few points relatively close to σ, and then we show that this implies that σ has all the desired properties.

Notation and terminology. Throughout the paper, $\log$ and $\exp$ are the base 2 logarithm and exponentiation, unless indicated otherwise. Let P be a set of n points in $\mathbb{R}^d$. We define a *separator* to be the boundary of an axis-aligned hypercube. A separator σ partitions $\mathbb{R}^d$ into two regions: a region σ_{in} consisting of all points in $\mathbb{R}^d$ inside or on σ, and a region σ_{out} consisting of all points in $\mathbb{R}^d$ strictly outside σ. We define the *size* of a separator σ to be its edge length, and we denote it by $\text{size}(\sigma)$. For a separator σ and a scaling factor $t \geqslant 0$, we define $t\sigma$ to be the separator obtained by scaling σ by a factor t with respect to its center. In other words, $t\sigma$ is the separator whose center is the same as the center of σ and with $\text{size}(t\sigma) = t \cdot \text{size}(\sigma)$; see Fig. 1(i).

A separator σ induces a partition of the given point set P into two subsets, $P \cap \sigma_{\text{in}}$ and $P \cap \sigma_{\text{out}}$. We are interested in δ-*balanced separators*, which are separators such that $\max(|P \cap \sigma_{\text{in}}|, |P \cap \sigma_{\text{out}}|) \leqslant \delta n$ for a fixed constant $\delta < 0$. It will be convenient to work with δ-balanced separators for $\delta = 4^d/(4^d + 1)$. From

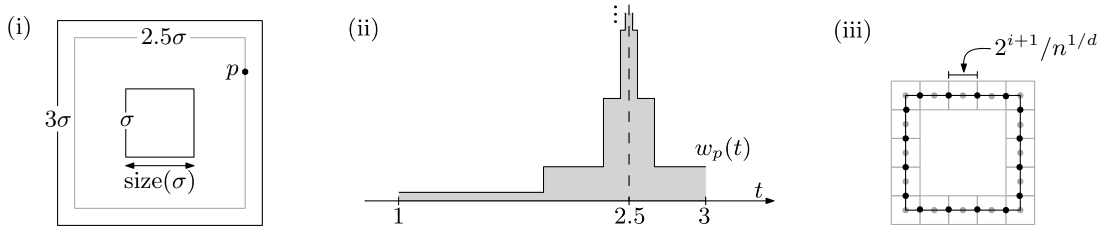

Fig. 1: (i) A separator σ and a point p with $\mathrm{rdist}(p, \sigma) = 0.75$. (ii) Schematic drawing of the weight function $w_p(t)$ of a point p such that $\mathrm{rdist}(p, t\sigma^*) = 0$ for $t = 2.5$. (iii) The grid G_i used in the proof of Theorem 5. The grid points are shown in black and gray; only the hypercubes H_g of the gray grid points are shown.

now on we will refer to $(4^d/(4^d+1))$-balanced separators simply as *balanced separators*. (There is nothing special about the constant $4^d/(4^d+1)$, and it could be made smaller by a more careful reasoning and at the cost of some other constants we will encounter later on.)

Distance-based separators for point sets. As mentioned, we first construct a separator σ such that there are only few points close to it. To this end we define the *relative distance* from a point p to σ, denoted by $\mathrm{rdist}(p, \sigma)$, as follows:

$$\mathrm{rdist}(p, \sigma) := d_\infty(p, \sigma)/\operatorname{size}(\sigma),$$

where $d_\infty(p, \sigma)$ denotes the shortest distance in the ℓ_∞-metric between p and any point on σ. Note that if t is the scaling factor such that $p \in t\sigma$, then $\mathrm{rdist}(p, \sigma) = |1 - t|/2$. For integers i define

$$P_i(\sigma) := \{\, p \in P \;:\; \mathrm{rdist}(p, \sigma) \leqslant 2^i/n^{1/d} \,\}.$$

Note that the smaller i is, the closer to σ the points in $P_i(\sigma)$ are required to be. We now wish to find a separator σ such that the size of the sets $P_i(\sigma)$ decreases rapidly as i decreases.

Theorem 1. *Let P be a set of n points in $\mathbb{R}^d$. Then there is a balanced separator σ for P such that*

$$|P_i(\sigma)| \;=\; \begin{cases} O((3/2)^i n^{1-1/d}) & \text{for all } i < 0 \\ O(4^i n^{1-1/d}) & \text{for all } 0 \leqslant i \end{cases}$$

Moreover, such a separator can be found in $O(n^{d+1})$ time.

Proof. Let σ^* be a smallest separator such that $|P \cap \sigma^*_{\mathrm{in}}| \geqslant n/(4^d+1)$. We will show that there is a t^* with $1 \leqslant t^* \leqslant 3$ such that $t^*\sigma^*$ is a separator with the required properties.

First we claim that $t\sigma^*$ is balanced for all $1 \leqslant t \leqslant 3$. To see this, observe that for $t \geqslant 1$ we have

$$|P \cap (t\sigma^*)_{\mathrm{out}}| \leqslant |P \cap \sigma^*_{\mathrm{out}}| = n - |P \cap \sigma^*_{\mathrm{in}}| \leqslant n - n/(4^d+1) = (4^d/(4^d+1))n.$$

Moreover, for $t \leqslant 3$ we can cover $t\sigma^*_{\mathrm{in}}$ by 4^d hypercubes of size at most $(3/4) \cdot \operatorname{size}(\sigma^*)$. By definition of σ^* these hypercubes contain less than $n/(4^d+1)$ points each, so $|P \cap (t\sigma^*)_{\mathrm{in}}| < 4^d \cdot (n/(4^d+1))$, which finishes the proof of the claim.

It remains to prove that there is a t^* with $1 \leqslant t^* \leqslant 3$ such that $t^*\sigma^*$ satisfies the condition on the sizes of the sets $P_i(t^*\sigma^*)$. To this end we will define a weight function $w_p : [1, 3] \to \mathbb{R}$ for each $p \in P$. The idea is that the closer p is to $t\sigma^*$, the higher the value $w_p(t)$. An averaging argument will then show that there must be a t^* such that $\sum_{p \in P} w_p(t^*)$ is sufficiently small, from which it follows that $t^*\sigma^*$ satisfies the condition on the sizes of the sets $P_i(t^*\sigma^*)$. Next we make this idea precise.

Assume without loss of generality that $\text{size}(\sigma^*) = 1$. For a point $p \in P$, let $i_p(t)$ be the integer such that $2^{i_p(t)-1}/n^{1/d} < \text{rdist}(p, t\sigma^*) \leqslant 2^{i_p(t)}/n^{1/d}$, where $i_p(t) = -\infty$ if $\text{rdist}(p, t\sigma^*) = 0$. Note that $p \in P_i(t\sigma^*)$ if and only if $i_p(t) \leqslant i$. Now we define the weight function $w_p(t)$ as follows; see Fig. 1(ii).

$$w_p(t) := \begin{cases} \frac{n^{1/d}}{(3/2)^{i_p(t)}} & \text{if } i_p(t) < 0 \\ \frac{n^{1/d}}{4^{i_p(t)}} & \text{if } i_p(t) \geqslant 0 \\ \text{undefined} & \text{if } i_p(t) = -\infty \end{cases}$$

We now want to bound $\int_1^3 w_p(t)\mathrm{d}t$. Note that the function $w_p(t)$ may be undefined for at most one $t \in [1, 3]$, namely when there is a t in this range such that $\text{rdist}(p, t\sigma^*) = 0$. Formally we should remove such a t from the domain of integration. To avoid cluttering the notation we ignore this technicality and continue to write $\int_1^3 w_p(t)\mathrm{d}t$.

Claim. For each $p \in P$, we have $\int_1^3 w_p(t)\mathrm{d}t = O(1)$.

Proof of claim. Define $T_p(i) := \{t : 1 \leqslant t \leqslant 3 \text{ and } i_p(t) = i\}$. By definition of $i_p(t)$, the value $w_p(t)$ is constant over $T_p(i)$. We therefore want to bound $|T_p(i)|$, the sum of the lengths of the intervals comprising $T_p(i)$. Assume without loss of generality that the center of σ^* lies at the origin of $\mathbb{R}^d$. Then, depending on the position of p, either $\text{rdist}(p, t\sigma^*) = |p_x - t|/\text{size}(t\sigma^*)$ or $\text{rdist}(p, t\sigma^*) = |p_y - t|/\text{size}(t\sigma^*)$. Assume without loss of generality that the former is the case. Since $1 \leqslant t \leqslant 3$ and $\text{size}(\sigma^*) = 1$, we then have $\text{rdist}(p, t\sigma^*) \geqslant |p_x - t|/3$. Hence, for any $t \in T_p(i)$ we have $|p_x - t|/3 \leqslant 2^i/n^{1/d}$. This implies that $|T_p(i)| \leqslant 6 \cdot 2^i/n^{1/d}$ and so

$$\int_1^3 w_p(t)\mathrm{d}t = \sum_{i \geqslant 0} |T_p(i)| \cdot \frac{n^{1/d}}{4^i} + \sum_{i<0} |T_p(i)| \cdot \frac{n^{1/d}}{(3/2)^i} \leqslant 6 \cdot \sum_{i \geqslant 0} \left(\frac{1}{2}\right)^i + 6 \cdot \sum_{i<0} \left(\frac{4}{3}\right)^i = O(1). \square$$

The above claim implies that $\int_1^3 \left(\sum_{p \in P} w_p(t)\right) \mathrm{d}t = O(n)$. Hence there exists a $t^* \in [1, 3]$ such that $\sum_{p \in P} w_p(t^*) = O(n)$. Now consider a set $P_i(t^*\sigma^*)$ with $i \geqslant 0$. Each $p \in P_i(t^*\sigma^*)$ has $i_p(t^*) \leqslant i$ and so

$$|P_i(t^*\sigma^*)| \leqslant \frac{\sum_{p \in P} w_p(t^*)}{\min_{p \in P_i(t^*\sigma^*)} w_p(t^*)} = \frac{O(n)}{n^{1/d}/4^i} = O(4^i n^{1-1/d}).$$

A similar argument shows that $|P_i(t^*\sigma^*)| = O((3/2)^i n^{1-1/d})$ for all $i < 0$.

To find the desired separator we first compute σ^*. Note that we can always shift σ^* such that it has at least one point on at least d of its $(d-1)$-dimensional faces. Hence, a simple brute-force algorithm can find σ^* in $O(n^{d+1})$ time. Once we have σ^*, we would like to find the value $t^* \in [1, 3]$ minimizing $\sum_{p \in P} w_p(t)$. Recall that each w_p is a step function, and so $\sum_{p \in P} w_p$ is a step function as well. There is one slight issue, however, namely that the number of steps of the functions w_p is unbounded. We deal with this issue by replacing each w_p by a truncated version $\overline{w}_p$, as explained next.

Note that the above arguments imply that there is a constant c_1 such that $\sum_{p \in P} w_p(t^*) < c_1 n$. We now define the truncated function $\overline{w}_p$ as follows: we set $\overline{w}_p(t) := 1/n$ if $w_p(t) < 1/n$, we set $\overline{w}_p(t) := c_1 n$ if $w_p(t) > c_1 n$, and we set $\overline{w}_p(t) := w_p(t)$ otherwise. Each function $\overline{w}_p$ is a step function, and one easily verifies that $\overline{w}_p$ has $O(\log n)$ steps which we can compute in $O(\log n)$ time. Hence, we can find a value $\overline{t}$ that minimizes $\sum_{p \in P} \overline{w}_p(t)$ in $O(n \log n)$ time. Since $\sum_{p \in P} w_p(\overline{t}) = O(n)$ if $\sum_{p \in P} \overline{w}_p(\overline{t}) = O(n)$, the separator $\overline{t}\sigma^*$ has the required properties. $\square$

Remark 2. It is not hard to speed up the time needed to compute the separator by working with an approximation of the smallest hypercube σ^* containing at least $n/(4^d + 1)$ points. Nonetheless, in our application this does not make a difference, and the simple brute-force algorithm to find σ^* suffices.

In the remainder we will need a slightly more general version of Theorem 1, where we require the separator to be balanced with respect to a given subset $Q \subseteq P$, that is, we require $\max(|Q \cap \sigma_{\rm in}|, |Q \cap \sigma_{\rm out}|) \leqslant \delta|Q|$ for $\delta = 4^d/(4^d+1)$. Note that the distance condition in the corollary below is still with respect to the points in P. The proof of the corollary is exactly the same as before, we only need to redefine σ^* to be a smallest separator such that $|Q \cap \sigma^*_{\rm in}| \geqslant |Q|/(4^d+1)$.

Corollary 3. *Let P be a set of n points in $\mathbb{R}^d$ and let $Q \subseteq P$. Then there is a separator σ that is balanced with respect to Q and such that*

$$|P_i(\sigma)| = \begin{cases} O((3/2)^i n^{1-1/d}) & \text{for all } i < 0 \\ O(4^i n^{1-1/d}) & \text{for all } 0 \leqslant i \end{cases}$$

Moreover, such a separator can be found in $O(n^{d+1})$ time.

A separator for TSP. Let P be a set of n points in $\mathbb{R}^d$, and let $\mathcal{S}(P)$ be the set of segments defined by P, that is, $\mathcal{S}(P) := \{pq : (p,q) \in P \times P\}$. Now consider a segment $s \in \mathcal{S}(P)$ and a separator σ. We say that s *crosses* σ if one endpoint of s lies in $\sigma_{\rm in}$ while the other lies in $\sigma_{\rm out}$. Using our distance-based separator for points we want to find a separator that is crossed only a few times by an optimal TSP tour. Moreover, we want to control the number of ways in which we have to "guess" a set of crossing segments. For this we will need the following crucial property of the segments in an optimal TSP tour.

Definition 4. *A set S of segments in $\mathbb{R}^d$ has the* packing property *if for any separator σ we have*

- *Packing Property (PP1):* $\big|\{s \in S : s \text{ crosses } \sigma \text{ and } \mathrm{length}(s) \geqslant \mathrm{size}(\sigma)\}\big| = O(1)$
- *Packing Property (PP2):* $\big|\{s \in S : s \subset \sigma_{\rm in} \text{ and } \mathrm{size}(\sigma)/4 \leqslant \mathrm{length}(s)\}\big| = O(1)$.

Property (PP2) is actually implied by (PP1), but it will be convenient to explicitly state (PP2) as part of the definition. Note that the constants hidden in the O-notation in Definition 4 may (and do) depend on d.

Some variants of the above packing property have been shown to hold for the set of edges of an optimal TSP [14, 23]. (For completeness, in this paper a proof in a more general setting can be found in Section 4.) Hence, we can restrict our attention to subsets of $\mathcal{S}(P)$ with the packing property. For a separator σ, we are thus interested in the following collection of sets of segments crossing σ:

$$\mathcal{C}(\sigma, P) := \{S \subseteq \mathcal{S}(P) : \ S \text{ has the packing property and all segments in } S \text{ cross } \sigma\}.$$

Our main separator theorem states that we can find a separator σ that is balanced and such that the sets in $\mathcal{C}(\sigma, P)$, as well as the collection $\mathcal{C}(\sigma, P)$ itself, are small. Since the general packing property is hard to test, in practice we can only enumerate a slightly larger collection of *candidate sets*, which we denote by $\mathcal{C}'(\sigma, P)$

Theorem 5. *Let P be a set of n points in $\mathbb{R}^d$ and let $Q \subseteq P$. Then there is a separator σ such that*

(i) *σ is balanced with respect to Q*
(ii) *each candidate set $S \in \mathcal{C}'(\sigma, P)$ contains $O(n^{1-1/d})$ segments*
(iii) $|\mathcal{C}(\sigma, P)| \leqslant |\mathcal{C}'(\sigma, P)| = 2^{O(n^{1-1/d})}$.

Moreover, σ and the collection $\mathcal{C}'(\sigma, P)$ can be computed in $2^{O(n^{1-1/d})}$ time.

Proof. Let σ be the separator obtained by applying Corollary 3 to the sets P and Q. Then σ has property (i). Next we prove that it has properties (ii) and (iii) as well, where we assume without loss of generality that $\mathrm{size}(\sigma) = 1$ and that σ is centered at the origin.

Let $L_{\rm small} := 1/(n^{1/d} n^{(1-1/d)\log_{3/2} 2})$. Any set $S \in \mathcal{C}(\sigma, P)$ can be partitioned into three subsets:

- $S_{\text{short}} := \{s \in S : \text{length}(s) \leqslant L_{\text{small}}\}$
- $S_{\text{mid}} := \{s \in S : L_{\text{small}} < \text{length}(s) \leqslant 1\}$
- $S_{\text{long}} := \{s \in S : \text{length}(s) > 1\}$.

We analyze these subsets separately. We start with S_{short} and S_{long}.

Claim. For any $S \in \mathcal{C}(\sigma, P)$ the set S_{short} consists of $O(1)$ segments, and the number of different subsets S_{short} that can arise over all sets $S \in \mathcal{C}(\sigma, P)$ is $n^{O(1)}$. Similarly, S_{long} consists of $O(1)$ segments, and the number of different subsets S_{long} that can arise over all sets $S \in \mathcal{C}(\sigma, P)$ is $n^{O(1)}$.

Proof of claim. A segment in S_{short} has both endpoints at distance at most L_{small} from σ, and so both endpoints are in $P_i(\sigma)$ for $i = -\log_{3/2} n^{1-1/d}$. By Corollary 3, the number of points in this $P_i(\sigma)$ is $O((3/2)^i n^{1-1/d}) = O(1)$. Hence, $|S_{\text{short}}| = O(1)$, which trivially implies we can choose S_{short} in $n^{O(1)}$ ways. The number of segments in S_{long} is $O(1)$ by Packing Property (PP1), which again implies that we can choose S_{long} in $n^{O(1)}$ ways. $\square$

It remains to handle S_{mid}.

Claim. For any $S \in \mathcal{C}(\sigma, P)$ the set S_{mid} consists of $O(n^{1-1/d})$ segments, and the number of different subsets S_{mid} that can arise over all sets $S \in \mathcal{C}(\sigma, P)$ is $2^{O(n^{1-1/d})}$.

Proof of claim. Define $S_{\text{mid}}(i) \subseteq S_{\text{mid}}$ to be the set of segments $s \in S_{\text{mid}}$ with $2^{i-1}/n^{1/d} < \text{length}(s) \leqslant 2^i/n^{1/d}$. Note that $S_{\text{mid}} = \bigcup\{S_{\text{mid}}(i) : -\log_{3/2} n^{1-1/d} + 1 \leqslant i \leqslant \log n^{1/d}\}$. We first analyze $|S_{\text{mid}}(i)|$ and the number of ways in which we can choose $S_{\text{mid}}(i)$ for a fixed i. To this end, we partition each face f of σ into a $(d-1)$-dimensional grid whose cells have size $2^i/n^{1/d}$. (If $n^{1/d}/2^i$ is not an integer, then we have size $1/\lceil n^{1/d}/2^i \rceil$; all subsequent arguments work in this case as well.) Let G_i be the set of grid points generated over all faces f, and note that $|G_i| = O((n^{1/d}/2^i)^{d-1}) = O(n^{1-1/d}/2^{i(d-1)})$. For each grid point $g \in G_i$, let H_g denote the axis-aligned hypercube of size $2^{i+1}/n^{1/d}$ centered at g; see Fig. 1(iii). Let $\mathcal{H}_i := \{H_g : g \in G_i\}$ be the set of all these hypercubes. Note that for any segment $s \in S_{\text{mid}}(i)$ there is a hypercube $H_g \in \mathcal{H}_i$ that contains s. Furthermore, all points in any H_g have distance at most $2^i/n^{1/d}$ from σ, and so $P \cap H_g \subseteq P_i(\sigma)$ for all g.

Now let n_g denote the number of points from P inside H_g. Since each point $p \in P$ is contained in a constant (depending on d) number of hypercubes H_g, we have $\sum_{g \in G_i} n_g = O(|P_i(\sigma)|)$. Furthermore, by Packing Property (PP2) we know that a hypercube H_g can contain only $O(1)$ segments from $S_{\text{mid}}(i)$. Thus

$$\begin{aligned} |S_{\text{mid}}(i)| &= O(\text{number of non-empty hypercubes } H_g) \\ &= O(\min(|G_i|, |P_i(\sigma)|)) \;=\; O(\min(n^{1-1/d}/2^{i(d-1)}, |P_i(\sigma)|)). \end{aligned}$$

For $i < 0$ we have $|P_i(\sigma)| = O((3/2)^i n^{1-1/d})$, which implies

$$|S_{\text{mid}}| = \sum_i |S_{\text{mid}}(i)| = O\left(\sum_{i<0} (3/2)^i n^{1-1/d} + \sum_{i \geqslant 0} n^{1-1/d}/2^{i(d-1)} \right) = O(n^{1-1/d}).$$

Now consider the number of ways in which we can choose $S_{\text{mid}}(i)$. In a hypercube H_g we have $O(1)$ edges which we can choose in $n_g^{O(1)}$ ways. Hence, if $G_i^* \subseteq G_i$ denotes the collection of grid points g such that $n_g > 0$ (in other words, such that H_g is non-empty), then

$$\text{total number of ways to choose } S_{\text{mid}}(i) \;=\; \prod_{g \in G_i^*} n_g^{O(1)} = 2^{O(\sum_{g \in G_i^*} \log n_g)}.$$

We bound $\sum_{g\in G_i^*} \log n_g$ separately for $i \geqslant 0$ and $i < 0$.

First consider the case $i \geqslant 0$. Here we have $\sum_{g\in G_i^*} n_g = O(|P_i(\sigma)|) = O(4^i n^{1-1/d})$. Moreover, $|G_i^*| \leqslant |G_i| = O(n^{1-1/d}/2^{i(d-1)})$ and $n^{1-1/d}/2^{i(d-1)} \leqslant 4^i n^{1-1/d}$, therefore there exists a constant c such that $|G_i^*| < c|P_i(\sigma)|$ for all $i \geqslant 0$. Hence,

$$\begin{aligned}\sum_{g\in G_i^*} \log n_g \leqslant |G_i^*| \cdot \log\left(\frac{|P_i(\sigma)|}{|G_i^*|}\right) &< |G_i^*| \cdot \log\left(\frac{c\cdot e\cdot |P_i(\sigma)|}{|G_i^*|}\right)\\ = O\left(\frac{n^{1-1/d}}{2^{i(d-1)}}\cdot \log 2^{i(d+1)}\right) &= O\left(\frac{i(d+1)}{2^{i(d-1)}}\cdot n^{1-1/d}\right)\end{aligned}$$

where the first step follows from the AM-GM inequality, and the third from the fact that $x\log(c\cdot e\cdot |P_i(\sigma)|/x)$ is monotone increasing for $x \in (0, c|P_i(\sigma)|)$; therefore we can replace $|G_i^*|$ with $|G_i|$ (since $|G_i^*| < |G_i| < c|P_i(\sigma)|$).

Now consider the case $i < 0$. Here we have $\sum_{g\in G_i^*} n_g = O(|P_i(\sigma)|) = O((3/2)^i n^{1-1/d})$, and $n^{1-1/d}/2^{i(d-1)} > (3/2)^i n^{1-1/d}$, thus the number of points to distribute is smaller than the number of available hypercubes, and so $\sum_{g\in G_i^*} \log n_g$ is maximized when $n_g = 2$ for all g (except for at most one grid point g). Hence,

$$\sum_{g\in G_i^*} \log n_g = O(|P_i(\sigma)|) = O((3/2)^i n^{1-1/d}).$$

Thus the total number of ways in which we can choose S_{mid} is bounded by the following expression, where i ranges from $i_{\min} := -\log_{3/2} n^{1-1/d} + 1$ to $i_{\max} := \log n^{1/d}$:

$$\begin{aligned}\prod_i \exp\left(O\left(\sum_{g\in G_i^*}\log n_g\right)\right) &= \prod_{i<0}\exp\Big(O\Big((3/2)^i n^{1-1/d}\Big)\Big)\prod_{i\geqslant 0}\exp\left(O\left(\frac{i(d+1)}{2^{i(d-1)}}\cdot n^{1-1/d}\right)\right)\\ &= \exp\left(O\left(\sum_{i<0}(3/2)^i n^{1-1/d} + \sum_{i\geqslant 0}\frac{i(d+1)}{2^{i(d-1)}}\cdot n^{1-1/d}\right)\right)\\ &= 2^{O(n^{1-1/d})}.\end{aligned}$$

□

Properties (ii) and (iii) now follow directly from the two claims above. Indeed, for any $S \in \mathcal{C}(\sigma, P)$ we have $|S| = |S_{\mathrm{short}}| + |S_{\mathrm{mid}}| + |S_{\mathrm{long}}| = O(1) + O(n^{1-1/d}) + O(1) = O(n^{1-1/d})$, and $|\mathcal{C}(\sigma, P)|$, which is the number of ways in which we can choose S, is $n^{O(1)} \cdot 2^{O(n^{1-1/d})} \cdot n^{O(1)} = 2^{O(n^{1-1/d})}$.

Notice that the above counting argument is constructive; we can also enumerate these sets in $2^{O(n^{1-1/d})}$ time; the enumerated collection $\mathcal{C}'(\sigma, P)$ has size $2^{O(n^{1-1/d})}$ and it is clearly a superset of $\mathcal{C}(\sigma, P)$. □

3 An exact algorithm for TSP

In this section, we design an exact algorithm for TSP using the separator theorem from the previous section. As a preliminary, let us take a look at the TSP problem in $\mathbb{R}^2$. The separator theorem from the previous section provides us with a separator σ such that the set of segments from an optimal tour that cross σ is $O(\sqrt{n})$. Moreover, the number of candidate subsets $S \in \mathcal{C}(\sigma, P)$ that we need to try is only $2^{O(\sqrt{n})}$. We can now obtain a divide-and-conquer algorithm similar to the algorithms of [14, 23] in a relatively standard manner. As we shall see, however, the resulting algorithm would still not run in $2^{O(\sqrt{n})}$ time. We will therefore need to modify the algorithm and employ the so-called rank-based approach [3] to get our final result. In what follows, we describe an exact algorithm for TSP in $\mathbb{R}^d$.

A separator-based divide-and-conquer algorithm for EUCLIDEAN TSP works as follows. We first compute a separator using Theorem 5 for the given point set. For each candidate subset of edges crossing the separator, we then need to solve a subproblem for the points inside the separator and one for the points outside the separator. In these subproblems we are no longer searching for a shortest tour, but for a collection of paths that connect the edges crossing the separator in a suitable manner. To define the subproblems more precisely, let P be a point set and let M be a perfect matching on a set $B \subseteq P$ of so-called *boundary points*. We say that a collection $\mathcal{P} = \{\pi_1, \ldots, \pi_{|B|/2}\}$ of paths *realizes M on P* if (i) for each pair $(p, q) \in M$ 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 length of a path π_i to be the sum of the Euclidean lengths of its edges, and we define the total length of $\mathcal{P}$ to be the sum of the lengths of the paths $\pi_i \in \mathcal{P}$. The subproblems that arise in our divide-and-conquer algorithm can now be defined as follows.

EUCLIDEAN PATH COVER
Input: A point set $P \subset \mathbb{R}^d$, a set of boundary points $B \subseteq P$, and a perfect matching M on B.
Question: Find a collection of paths of minimum total length that realizes M on P.

Note that we can solve EUCLIDEAN TSP on a point set P by creating a copy p' of an arbitrary point $p \in P$, and then solve EUCLIDEAN PATH COVER on $P \cup \{p'\}$ with $B := \{p, p'\}$ and $M := \{(p, p')\}$.

A generic instance of EUCLIDEAN PATH COVER can be solved by a separator-based recursive algorithm as follows. Let σ be a separator for P. To solve EUCLIDEAN PATH COVER for the input (P, B, M), we consider each candidate set $S \in \mathcal{C}(\sigma, P)$ of edges crossing the separator σ. In fact, it is sufficient to consider candidate sets where the number of segments from S incident to any point $p \in P \setminus B$ is at most two, and the number of segments from S incident to any point in B is at most one. We now wish to define subproblems for σ_{in} and σ_{out} (the regions inside and outside σ, respectively) whose combination yields a solution for the given problem on P. Let $P_1(S) \subseteq P$ denote the set of endpoints with precisely one incident segment from S, and let $P_2(S) \subseteq P$ be the set of endpoints with precisely two incident segments from S. Note that in a solution to the problem (P, B, M) the points in B need one incident edge—they must become endpoints of a path—while points in $P \setminus B$ need two incident edges. This means that the points in $B \cap P_1(S)$ and the points in $P_2(S)$ now have the desired number of incident edges, so they can be ignored in the subproblems. Points in $B \triangle P_1(S) := (B \setminus P_1(S)) \cup (P_1(S) \setminus B)$ still need one incident edge, while points in $P \setminus ((B \cap P_1(S)) \cup P_2(S))$ still need two incident edges. Hence, for σ_{in} we obtain subproblems of the form $(P_{\text{in}}, B_{\text{in}}, M_{\text{in}})$ where

$$\begin{aligned} P_{\text{in}} &:= \Big(P \setminus \big((B \cap P_1(S)) \cup P_2(S)\big)\Big) \cap \sigma_{\text{in}}, \\ B_{\text{in}} &:= \big(B \triangle P_1(S)\big) \cap \sigma_{\text{in}}, \\ M_{\text{in}} & \text{ is a perfect matching on } B_{\text{in}}. \end{aligned} \tag{1}$$

See Figure 2. For σ_{out} we obtain subproblems of the form $(P_{\text{out}}, B_{\text{out}}, M_{\text{out}})$ defined in a similar way. As already remarked, we can restrict our attention to candidate sets $S \in \mathcal{C}(\sigma, P)$ that contain at most one edge incident to any given point in B, and at most two edges incident to any given point in $P \setminus B$. Moreover, S should be such that $|B_{\text{in}}|$ and $|B_{\text{out}}|$ are even. We define $\mathcal{C}^*(\sigma, P)$ to be the family of candidate sets $\mathcal{C}'(\sigma, P)$ restricted this way. Also note that while $P_{\text{in}}, B_{\text{in}}$ and $P_{\text{out}}, B_{\text{out}}$ are determined once S is fixed, the algorithm has to find the best matchings M_{in} and M_{out}. These matchings should together realize the matching M on P and, among all such matchings, we want the pair that leads to a minimum-length solution.

The number of perfect matchings on k points is $k^{\Theta(k)}$. Unfortunately, already in the first call of the recursive algorithm $|B_{\text{in}}|$ and $|B_{\text{out}}|$ can be as large as $\Theta(n^{1-1/d})$. Hence, recursively checking all matchings

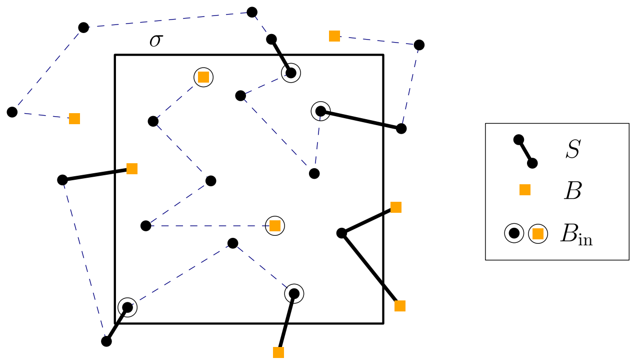

Fig. 2: A EUCLIDEAN PATH COVER instance, and how it is sliced by a separator σ. The dashed lines show a possible path cover.

will not lead to an algorithm with the desired running time. In $\mathbb{R}^2$ we can use that an optimal TSP tour is crossing-free, so it is sufficient to look for "crossing-free matchings", of which there are only $2^{O(k)}$. (This approach would actually require a different setup of the subproblems; see the papers by Deineko *et al.* [8] and Dorn *et al.* [9].) However, the crossing-free property has no analogue in higher dimensions, and it does not hold in $\mathbb{R}^2$ for our "almost-Euclidean" setting either. Hence, we need a different approach to rule out a significant proportion of the available matchings.

Applying the rank-based approach. Next we describe how we can use the rank-based approach [3, 6] in our setting. A standard application of the rank-based approach works on a tree-decomposition of the underlying graph, where the bags represent vertex separators of the underlying graph. In our application the underlying graph is a complete graph on the points—all segments are potentially segments of the TSP tour—and we have to use a separator for the edges in the solution. We try to avoid the intricate notation introduced in the original papers, but our terminology is mostly compatible with [3].

Let P be a set of points in $\mathbb{R}^d$, and let $B \subseteq P$ be a set of boundary points such $|B|$ is even. 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 solution realizing M. In other words, $\text{weight}(M)$ is the length of the solution of EUCLIDEAN PATH COVER for input (P, B, M). Whenever we speak of weighted matchings in the sequel, we always mean perfect matchings on a set $B \subseteq P$ weighted as above, where B and P should be clear from the context. We use $\mathcal{M}(B, P)$ to denote the set of all such weighted matchings on B. Note that $|\mathcal{M}(B, P)| = |\mathcal{M}(B)| = 2^{O(|B| \log |B|)}$. The key to reducing the number of matchings we have to consider is the concept of representative sets, as explained next.

We say that two matchings $M, M' \in \mathcal{M}(B)$ *fit* if their union is a Hamiltonian cycle. 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 $\text{opt}(M, \mathcal{R}) := \min\{\text{weight}(M') : M' \in \mathcal{R}, M' \text{ fits } M\}$, 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 $\mathcal{R} \subseteq \mathcal{M}(B, P)$ of weighted matchings is defined to be *representative* of another set $\mathcal{R}' \subseteq \mathcal{M}(B, P)$ 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 $\mathcal{M}(B, P)$, because it is also restricted by the Packing Property, while a solution of Euclidean Path Cover for a generic P, B, M may not satisfy it. Let $\mathcal{M}'(B, P)$ denote the set of weighted matchings in $\mathcal{M}(B, P)$ that have a corresponding EUCLIDEAN PATH COVER solution satisfying the Packing Property.

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

Lemma 6. [Bodlaender *et al.* [3], Theorem 3.7] *There exists a set $\mathcal{R}^*$ consisting of $2^{|B|-1}$ weighted matchings that is representative of the set $\mathcal{M}(B, P)$. Moreover, there is an algorithm* Reduce *that, given a representative set $\mathcal{R}$ of $\mathcal{M}(B, P)$, computes such a set $\mathcal{R}^*$ in $|\mathcal{R}| \cdot 2^{O(|B|)}$ time.*

Lemma 6 can also be applied for our case, where $\mathcal{R}$ is representative of $\mathcal{M}'(B, P) \subseteq \mathcal{M}(B, P)$, the set of weighted matchings in $\mathcal{M}(B, P)$ that have a corresponding EUCLIDEAN PATH COVER solution satisfying the Packing Property. The result of Bodlaender *et al.* is actually more general than stated above, as it not only applies to matchings but also to other types of partitions. Moreover, for matchings the bound has been improved to $2^{|B|/2-1}$ [6]. However, Lemma 6 suffices for our purposes.

Lemma 6 bounds the size of the representative set in terms of $|B|$, the number of boundary points. In the first call of our algorithm $|B| = O(n^{1-1/d})$ because of the properties of our separator, but we have to be careful that the size of B stays under control in recursive calls. A key step in Algorithm 1, which describes the global working of our algorithm, is therefore Step 4, where we invoke the balance condition of the separator with respect to B or P depending on the size of B relative to the size of P. (The constant γ will be specified in the analysis of the running time.)

Algorithm 1 *TSP-Repr*(P, B)

Input: A set P of points in $\mathbb{R}^d$ and a subset $B \subseteq P$
Output: A set $\mathcal{R}$ of weighted partitions of size at most $2^{|B|-1}$ that represents $\mathcal{M}'(B, P)$

```
1:  if |P| ⩽ 1 then R ← {(∅, 0)}
2:  else
3:      R ← ∅
4:      Compute a separator σ using Theorem 5, where Q = P if |B| ⩽ γ|P|^{1−1/d} and Q = P otherwise.
5:      for all candidate set S ∈ C*(σ, P) do
6:          Set R_in ← TSP-Repr(P_in, B_in), where P_in and B_in are defined according to (1)
7:          Set R_out ← TSP-Repr(P_out, B_out), where P_out and B_out are defined according to (1)
8:          for all combinations of weighted matchings M_in ∈ R_in and M_out ∈ R_out do
9:              if M_in and M_out are compatible then
10:                 Insert Join_S(M_in, M_out) into R with weight weight(M_in) + length(S) + weight(M_out)
11:     R ← Reduce(R)
12: return R
```

It remains to explain how we combine the representative sets $\mathcal{R}_{\text{in}}$ and $\mathcal{R}_{\text{out}}$ in Steps 8–Steps 11.

Consider a set $S \in \mathcal{C}^*(\sigma, P)$, a matching $M_{\text{in}} \in \mathcal{M}(B_{\text{in}})$ and a matching $M_{\text{out}} \in \mathcal{M}(B_{\text{out}})$. Let $G = G_S(M_{\text{in}}, M_{\text{out}})$ be the graph with vertex set $V(G) := B \cup P_1(S) \cup P_2(S)$ and edge set $E(G) := M_{\text{in}} \cup M_{\text{out}} \cup S$. We say that M_{in} and M_{out} are *compatible* if G consists of $|B|/2$ disjoint paths covering $V(G)$ whose endpoints are exactly the points in B. A pair of compatible matchings induces a perfect matching on B, where for each of these $|B|/2$ paths we add a matching edge between its endpoints. We denote this matching by $\text{Join}_S(M_{\text{in}}, M_{\text{out}}) \in \mathcal{M}(B)$. To get a set $\mathcal{R}$ of weighted matchings on B we thus iterate in Steps 8–10 through all pairs $M_{\text{in}}, M_{\text{out}}$ where M_{in} and M_{out} are compatible, and for such pairs, we add to $\mathcal{R}$ the matching $\text{Join}_S(M_{\text{in}}, M_{\text{out}})$. The weight of this matching is $\text{weight}(M_{\text{in}}) + \text{length}(S) + \text{weight}(M_{\text{out}})$.

Claim. The set $\mathcal{R}$ constructed in Lines 5–10 of Algorithm 1 is representative of $\mathcal{M}'(B, P)$.

Proof of claim. The proof is by induction on $|P|$. Clearly, for $|P| \leqslant 1$ the claim holds. Otherwise, let $S \in \mathcal{C}^*(\sigma, P)$ be fixed. The set S is considered in some iteration of the outer loop. Define $P_{\text{in}}, P_{\text{out}}, B_{\text{in}}, B_{\text{out}}$ as in this iteration, and let $\mathcal{R}_{\text{in}}, \mathcal{R}_{\text{out}}$ be the sets returned by the recursive calls, which are representative sets of $\mathcal{M}'(B_{\text{in}}, P_{\text{in}})$ and $\mathcal{M}'(B_{\text{out}}, P_{\text{out}})$ respectively by induction. Notice that S can be regarded as a EUCLIDEAN PATH COVER solution for $(B_{\text{in}} \cup B_{\text{out}}, B_{\text{in}} \cup B_{\text{out}}, M_S)$, where M_S is the matching realized by S on $B_{\text{in}} \cup B_{\text{out}}$. Let $\text{length}(S)$ be the weight assigned to M_S. Clearly $\{M_S\}$ is representative of $\{M_S\}$. Now our Join_S operation can be regarded as the succession of two join operations as defined by [3], applied to $\{M_S\}$ and $\mathcal{R}_{\text{in}}$ first, and then to the result and $\mathcal{R}_{\text{out}}$ second. By Lemma 3.6 in [3], the join operation preserves representation, therefore the matchings added to $\mathcal{R}$ in this iteration of the outer loop form a representative set of

$$\widehat{\mathcal{M}}_S := \{\mathrm{Join}_S(M_{\mathrm{in}}, M_{\mathrm{out}}) \mid M_{\mathrm{in}} \in \mathcal{M}'(B_{\mathrm{in}}, P_{\mathrm{in}}), \\ M_{\mathrm{out}} \in \mathcal{M}'(B_{\mathrm{out}}, P_{\mathrm{out}})\}.$$

Consequently, the set $\mathcal{R}$ that is created at the end of the outer loop is a representative set of $\widehat{\mathcal{M}} := \bigcup_{S \in \mathcal{C}^*(P,\sigma)} \widehat{\mathcal{M}}_S$. The set $\widehat{\mathcal{M}}_S$ contains the subset of $\mathcal{M}'(B,P)$ that has a corresponding optimum with the Packing Property that intersects σ in S, because for any such optimum path cover $\mathcal{P}$, the subpaths of $\mathcal{P}$ induced by P_{in} also have the packing property, and form an optimal EUCLIDEAN PATH COVER for the input $(P_{\mathrm{in}}, B_{\mathrm{in}}, M_{\mathrm{in}})$, i.e., there is a corresponding weighted matching $M_{\mathrm{in}} \in \mathcal{M}'(B_{\mathrm{in}}, P_{\mathrm{in}})$. (The analogous statement is true for the subpaths induced by P_{out}.) Since $\mathcal{C}^*(\sigma, P)$ contains all sets S that can arise as the set of segments intersecting σ in an optimum EUCLIDEAN PATH COVER solution with the Packing Property, it follows that $\widehat{\mathcal{M}} \supseteq \mathcal{M}'(B,P)$ and the claim holds. □

Notice that $\mathcal{R}$ can be computed in a brute-force manner in $O(|\mathcal{R}_{\mathrm{in}}| \cdot |\mathcal{R}_{\mathrm{out}}| \cdot \mathrm{poly}(|B| + |S|))$ time. By combining $\mathcal{R}_{\mathrm{in}}$ and $\mathcal{R}_{\mathrm{out}}$ in this manner, the size of $\mathcal{R}$ may be more than $2^{|B|-1}$. Hence, we apply the *Reduce* algorithm [3], to create a representative set of size at most $2^{|B|-1}$ in $|\mathcal{R}| \cdot 2^{O(|B|)}$ time. Since our recursive algorithm ensures that $|\mathcal{R}_{\mathrm{in}}| \leqslant 2^{|B_{\mathrm{in}}|-1}$ and $|\mathcal{R}_{\mathrm{out}}| \leqslant 2^{|B_{\mathrm{out}}|-1}$, all of the above steps run in $2^{O(|B|+|S|)} = \exp\big(O\big(|B| + |P|^{1-1/d}\big)\big)$ time.

Analysis of the running time. The running time of *TSP-Repr*(P,B) essentially satisfies the following recurrence, where c_0, c_1, c_2 are positive constants and we use the notation $n := |P|$ and $b := |B|$.

$$T(n,b) \leqslant \begin{cases} c_0 & \text{if } n \leqslant 1 \\ 2^{c_1(n^{1-1/d}+b)} T\left(\delta n,\, b + c_2 n^{1-1/d}\right) & \text{if } b \leqslant \gamma n^{1-1/d} \\ 2^{c_1(n^{1-1/d}+b)} T\left(n,\, \delta b + c_2 n^{1-1/d}\right) & \text{if } b > \gamma n^{1-1/d}, \end{cases}$$

The actual recurrence is a bit more subtle—see Subsection 3.1 for a precise formula, and for a proof that the running time for the initial call is $T(n,2) = 2^{O(n^{1-1/d})}$.

Theorem 7. *For any fixed $d \geqslant 2$ there is an algorithm for* EUCLIDEAN TSP *in $\mathbb{R}^d$ that runs in $2^{O(n^{1-1/d})}$ time. Moreover, there is no $2^{o(n^{1-1/d})}$ algorithm for* EUCLIDEAN TSP *in $\mathbb{R}^d$, unless ETH fails.*

Proof. The upper bound follows from the discussion above. The lower bound is a direct consequence of a recent lower bound on HAMILTONIAN CYCLE in induced grid graphs [7], as explained next.

Let G_d be the d-dimensional grid in $\mathbb{R}^d$ whose cells have side length 1, and where all grid points have positive integral coordinates. An *induced graph graph* in $\mathbb{R}^d$ is a graph $\mathcal{G}$ whose vertex set is a set P of n grid points in G_d, specified by their coordinates, and where there is an edge between two vertices if and only if the Euclidean distance between the corresponding points in P is exactly 1. It is straightforward to verify that $\mathcal{G}$ has a Hamiltonian cycle if and only if P has a TSP tour of length n. Hence, the lower bound on EUCLIDEAN TSP follows from the $2^{\Omega(n^{1-1/d})}$ lower bound for HAMILTONIAN CYCLE in induced grid graphs, proved by De Berg *et al.* [7]. □

3.1 Detailed analysis of the running time

In this part, we provide a detailed analysis of the running time of our algorithm. We first derive a more precise recurrence relation than the intuitive one stated earlier, and then we solve the recurrence.

For each $S \in \mathcal{C}^*(\sigma, P)$, let $n_{S,\mathrm{in}} := |P_{\mathrm{in}}|$, let $b_{S,\mathrm{in}} := |B_{\mathrm{in}}|$, let $n_{S,\mathrm{out}} := |P_{\mathrm{out}}|$, and let $b_{S,\mathrm{out}} := |B_{\mathrm{out}}|$. By the discussion in Section 3, we can bound the running time of the two inner loops, the *Reduce* algorithm and the rest of the operations outside the recursive calls by $\exp(c_3(n^{1-1/d}+b))$ for some positive constant c_3. Therefore, the algorithm *TSP-Repr*(P,B) obeys the following recursion, where σ_P and σ_B are separators balanced with respect to P and B, respectively.

$$T(n,b) \leqslant \begin{cases} c_0 & \text{if } n \leqslant 1 \\ \sum_{S\in\mathcal{C}^*(\sigma_P,P)} \big(\exp(c_3(n^{1-1/d}+b)) + T(n_{S,\text{in}},b_{S,\text{in}}) + T(n_{S,\text{out}},b_{S,\text{out}})\big) & \text{if } b \leqslant \gamma n^{1-1/d} \\ \sum_{S\in\mathcal{C}^*(\sigma_B,P)} \big(\exp(c_3(n^{1-1/d}+b)) + T(n_{S,\text{in}},b_{S,\text{in}}) + T(n_{S,\text{out}},b_{S,\text{out}})\big) & \text{if } b > \gamma n^{1-1/d}. \end{cases}$$

Lemma 8. $T(n,2) = 2^{O(n^{1-1/d})}$.

Proof. We prove by induction that $T(n,b) \leqslant \exp(d_1 n^{1-1/d} + d_2 b)$ for some constants d_1 and d_2 and for all $1 \leqslant b \leqslant n$. This clearly holds for $b,n \leqslant 1$, so by induction, for each S we have

$$\begin{aligned} &\exp(c_3(n^{1-1/d}+b)) + T(n_{S,\text{in}},b_{S,\text{in}}) + T(n_{S,\text{out}},b_{S,\text{out}}) \\ &\qquad \leqslant \exp(c_3(n^{1-1/d}+b))\exp\left(d_1 n_{S,\text{in}}^{1-1/d} + d_2 b_{S,\text{in}} + d_1 n_{S,\text{out}} + d_2 b_{S,\text{out}}\right). \end{aligned}$$

Let c_2 and c_4 be the constants from part (ii) and (iii) of Theorem 5. For any $S \in \mathcal{C}^*(\sigma_B,P)$, we have $b_{S,\text{in}} \leqslant \delta b + c_2 n^{1-1/d}$ and $b_{S,\text{out}} \leqslant \delta b + c_2 n^{1-1/d}$; similarly, for any $S \in \mathcal{C}^*(\sigma_P,P)$, we have $n_{S,\text{in}} \leqslant \delta n$ and $n_{S,\text{out}} \leqslant \delta n$. In the remaining cases, we can just use the trivial bounds $b_{S,.} \leqslant b + c_2 n^{1-1/d}$ and $n_{S,.} \leqslant n$. Since $|\mathcal{C}^*(\sigma,P)| \leqslant \exp(c_4 n^{1-1/d})$, we get the following:

$$T(n,b) \leqslant \begin{cases} c_0 & \text{if } n \leqslant 1 \\ \exp(c_1(n^{1-1/d}+b))\exp\big(d_1(\delta n)^{1-1/d} + d_2(b + c_2 n^{1-1/d})\big) & \text{if } b \leqslant \gamma n^{1-1/d} \\ \exp(c_1(n^{1-1/d}+b))\exp\big(d_1 n^{1-1/d} + d_2(\delta b + c_2 n^{1-1/d})\big) & \text{if } b > \gamma n^{1-1/d}, \end{cases}$$

where $c_1 = c_3 + c_4$. We set $c := \max(c_1,c_2)$, and let $\gamma := \frac{2c}{1-\delta}$. (Notice that the definition of γ here is valid: it is independent of d_1 and d_2.)

If $b \leqslant \gamma n^{1-1/d} = \frac{2c}{1-\delta} n^{1-1/d}$, we have the following:

$$\begin{aligned} T(n,b) &\leqslant \exp(cn^{1-1/d} + cb)\exp\big(d_1(\delta n)^{1-1/d} + d_2 b + d_2 c n^{1-1/d}\big) \\ &\leqslant \exp\left(\left(c + \frac{2c}{1-\delta} + d_1\delta^{1-1/d} + d_2 c\right) n^{1-1/d} + d_2 b\right) \\ &\leqslant \exp(d_1 n^{1-1/d} + d_2 b), \end{aligned}$$

where the third inequality uses $b \leqslant \frac{2c}{1-\delta} n^{1-1/d}$ and the fourth uses

$$c + \frac{2c}{1-\delta} + d_1\delta^{1-1/d} + d_2 c \leqslant d_1 \;\Leftrightarrow\; d_1 \geqslant \frac{c + 2c/(1-\delta) + d_2 c}{1-\delta^{1-1/d}};$$

note that $\delta^{1-1/d} < 1$. Finally, if $b > \gamma n^{1-1/d} = \frac{2c}{1-\delta} n^{1-1/d}$, we have the following:

$$\begin{aligned} T(n,b) &\leqslant \exp(cn^{1-1/d} + cb)\exp\big(d_1 n^{1-1/d} + d_2\delta b + d_2 c n^{1-1/d}\big) \\ &< \exp\left(d_1 n^{1-1/d} + \left(\frac{1-\delta}{2} + c + \delta d_2 + \frac{1-\delta}{2} d_2\right) b\right) \\ &\leqslant \exp(d_1 n^{1-1/d} + d_2 b), \end{aligned}$$

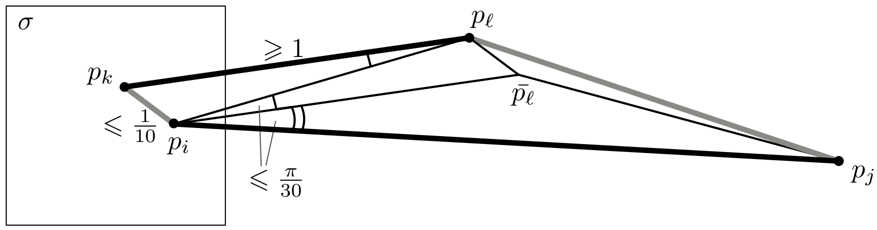

Fig. 3: Since $|p_ip_j| > |p_jp_\ell|$ and $|p_kp_\ell| > |p_ip_k|$, we can exchange p_ip_j and p_kp_ℓ for p_ip_k and p_jp_ℓ in the tour, and get a shorter tour. Note for $d > 2$ the points do not all have to lie in the same plane.

where the strict inequality uses $cn^{1-1/d} < \frac{1-\delta}{2}b$, and the final inequality uses

$$\frac{1-\delta}{2} + c + \frac{1+\delta}{2}d_2 \leqslant d_2 \;\Leftrightarrow\; d_2 \geqslant \frac{(1-\delta)/2 + c}{1-(1+\delta)/2};$$

note that $(1+\delta)/2 < 1$. Since there exists positive constants d_1 and d_2 satisfying the above inequalities, we have that $T(n,b) \leqslant \exp(d_1 n^{1-1/d} + d_2 b)$, and in particular, for the initial call we have $T(n,2) = 2^{O(n^{1-1/d})}$. □

4 Almost Euclidean TSP

So far we considered EUCLIDEAN TSP in the real-RAM model of computation. We now consider a slightly more general scenario in the Word-RAM model. Here we assume that the input is a set P of n points in $\mathbb{R}^d$, specified by their coordinates (which are rational in the Word-RAM model), as well as a distance matrix D. The basic assumption we make is that the distances in D approximate the real Euclidean distances well. More precisely, we require that the ordering of pairwise distances on the given point set $P := \{p_1, \dots, p_n\}$ is preserved: if $|p_ip_j| < |p_kp_\ell|$ then $D[i,j] < D[k,\ell]$. We call this the *almost Euclidean* version of TSP.

In order to show that our algorithms work in this setting, we only need to show that an optimal tour in this setting satisfies Packing Property. Note that the Packing Property for the almost Euclidean version immediately implies that the Packing Property also holds for the Euclidean version (where, as remarked in Section 2, similar properties were already known).

Theorem 9. *Let $P := \{p_1, \dots, p_n\}$ be a point set in $\mathbb{R}^d$ and let D be a distance matrix for P such that $|p_ip_j| < |p_kp_\ell| \iff D[i,j] < D[k,\ell]$. Let T be a tour on P that is optimal for the distances given by D. The the set of edges of T has the Packing Property.*

Proof. We first prove Packing Property (PP1) and then argue that (PP2) follows from (PP1).

Let σ be a hypercube, and suppose without loss of generality that $\mathrm{size}(\sigma) = 1$. Suppose for a contradiction that there are more than c tour edges of length at least 1 that cross σ, where c is a suitably large constant (which depends on d). By the pigeonhole principle, We can then find three edges in T such that (i) the pairwise Euclidean distances between the endpoints of these edges that lie inside σ_{in} is at most $1/10$, and (ii) the pairwise angles between these edges is at most $\pi/30$. (Here the angle between two edges is measured as the smaller angle between two lines going through the origin and parallel to the given edges.)

Now fix an orientation on the tour T such that at least two of the three edges cross σ from inside to outside, and orient these edges accordingly. Let p_ip_j and p_kp_ℓ denote these two oriented edges; see Figure 3. Thus $p_i, p_k \in \sigma_{\mathrm{in}}$ and $|p_ip_k| \leqslant 1/10$. Assume without loss of generality that $|p_ip_j| \geqslant |p_kp_\ell|$. In the triangle $p_ip_kp_\ell$, we have $|p_kp_\ell| \geqslant 1$ and $|p_ip_k| \leqslant 1/10$, therefore, $\sphericalangle(p_ip_\ell p_k) \leqslant \arcsin(1/10) < \pi/30$. Let $\bar{p}_\ell = p_i + (p_\ell - p_k)$. (Here and in the sequel we use that a point can also be thought of as a vector, so we can add and subtract points to get new points.) Then we have that $\sphericalangle(p_\ell p_i \bar{p}_\ell) = \sphericalangle(p_i p_\ell p_k) \leqslant \pi/30$. Due to our choice of $p_i p_j$ and $p_k p_\ell$, which ensures that their angle is at most $\pi/30$, we have

$$\sphericalangle(\bar{p}_\ell p_i p_j) = \sphericalangle\big((p_j - p_i), (p_\ell - p_k)\big) \leqslant \pi/30.$$

Therefore, $\sphericalangle(p_\ell p_i p_j) \leqslant \pi/15$. Note that this is also true if the points do not all lie in the same plane. Now observe that

$$|p_i p_\ell| \leqslant |p_k p_\ell| + |p_i p_k| \leqslant |p_i p_j| + 1/10,$$

that is, $|p_i p_\ell|$ cannot be much longer than $|p_i p_j|$. We also have $9/10 \leqslant |p_k p_\ell| - |p_i p_k| \leqslant |p_i p_\ell|$. Thus, if we look at the triangle $p_i p_\ell p_j$, then we have $9/10 \leqslant |p_i p_\ell| \leqslant |p_i p_j| + 1/10$, and $|p_i p_j| \geqslant 1$ and $\sphericalangle(p_\ell p_i p_j) \leqslant \pi/15$. Hence, $|p_i p_j| > |p_j p_\ell|$. Indeed, if $\sphericalangle(p_i p_\ell p_j) \geqslant \pi/2$ then we can immediately conclude that $|p_i p_j|$ is the longest side of the triangle $p_i p_\ell p_j$; otherwise $p_i p_\ell$ and $p_i p_j$ have roughly equal lengths, and since $\sphericalangle(p_\ell p_i p_j) \leqslant \pi/15$ we get that $|p_i p_j| > |p_j p_\ell|$. Moreover, we obviously have $|p_k p_\ell| > |p_i p_k|$ since $|p_k p_\ell| \geqslant 1$ and $|p_i p_k| \leqslant 1/10$.

Because the ordering of the pairwise distances in the matrix D is the same as for the Euclidean distances, we can conclude that $D[i,j] > D[j,\ell]$ and $D[k,\ell] > D[i,k]$. But then we can exchange $p_i p_j$ and $p_k p_\ell$ for $p_i p_k$ and $p_j p_\ell$ in the tour T—because both edges are oriented from inside σ to outside σ this gives a valid tour—and get a shorter tour. This contradicts the minimality of the tour, concluding the proof of (PP1).

Property (PP2) is a direct consequence of (PP1). Indeed, if we cover σ_{in} by $O(1)$ hypercubes of diameter $\text{size}(\sigma_{\text{in}})/5$, then any segment of length at least $\text{size}(\sigma_{\text{in}})/4$ inside σ_{in} crosses at least one such hypercube, and by (PP1) each hypercube is crossed by $O(1)$ edges of length at least $\text{size}(\sigma_{\text{in}})/4$. $\square$

Remark 10. It would be useful for applications if the algorithm could work with a distance matrix that is a constant distortion of the Euclidean distances, that is, a matrix D such that $(1/\alpha) \cdot |p_i p_j| \leqslant D[i,j] \leqslant \alpha |p_i p_j|$ for some constant $\alpha \geqslant 1$. Unfortunately, while (PP2) holds also in this scenario, (PP1) does not.

5 Concluding remarks

In this paper we described a new geometric separation technique, which resulted in a faster exact algorithm for EUCLIDEAN TSP. We also showed that this algorithm is tight unless ETH fails, thus settling the complexity of EUCLIDEAN TSP (assuming ETH, and up to the constant in the exponent).

We believe that our separation technique can be useful for other problems in Euclidean geometry as well, and in particular for problems where one wishes to compute a minimum-length geometric structure that satisfies the Packing Property. An example of such a problem is RECTILINEAR STEINER TREE. An additional issue to overcome here is that the number of potential Steiner points is $O(n^d)$, which means that a direct application of our techniques does not work. Another challenging problem is finding the minimum weight triangulation for a set of n points given in $\mathbb{R}^2$, which was proven NP-hard by Mulzer and Rote [19] and for which an $n^{O(\sqrt{n})}$ algorithm is known [16]. A minimum-weight triangulation does not have the packing property, because of clusters of points that are far from each other, but finding an optimal triangulation between such clusters can possibly be handled separately.

References

[1] S. Arora. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM*, 45(5):753–782, 1998.

[2] R. Bellman. Dynamic programming treatment of the travelling salesman problem. *J. ACM*, 9(1):61–63, 1962.

[3] H. L. Bodlaender, M. Cygan, S. Kratsch, and J. Nederlof. Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. *Information and Computation*, 243:86–111, 2015.

[4] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

[5] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

[6] M. Cygan, S. Kratsch, and J. Nederlof. Fast Hamiltonicity checking via bases of perfect matchings. *J. ACM*, 65(3):12:1–12:46, Mar. 2018.

[7] M. de Berg, H. L. Bodlaender, S. Kisfaludi-Bak, D. Marx, and T. C. van der Zanden. A framework for ETH-tight algorithms and lower bounds in geometric intersection graphs. In *STOC*, pages 574–586. ACM, 2018.

[8] V. G. Deineko, B. Klinz, and G. J. Woeginger. Exact algorithms for the Hamiltonian cycle problem in planar graphs. *Oper. Res. Lett.*, 34(3):269–274, 2006.

[9] F. Dorn, E. Penninkx, H. L. Bodlaender, and F. V. Fomin. Efficient exact algorithms on planar graphs: Exploiting sphere cut decompositions. *Algorithmica*, 58(3):790–810, 2010.

[10] M. R. Garey, R. L. Graham, and D. S. Johnson. Some NP-complete geometric problems. In *STOC*, pages 10–22. ACM, 1976.

[11] M. Held and R. M. Karp. A dynamic programming approach to sequencing problems. In *Proceedings of the 1961 16th ACM National Meeting*, ACM '61, pages 71.201–71.204, New York, NY, USA, 1961. ACM.

[12] R. Z. Hwang, R. C. Chang, and R. C. T. Lee. The searching over separators strategy to solve some NP-hard problems in subexponential time. *Algorithmica*, 9(4):398–423, 1993.

[13] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[14] V. Kann. *On the approximability of NP-complete optimization problems*. PhD thesis, Royal Institute of Technology Stockholm, 1992.

[15] R. M. Karp. Reducibility among combinatorial problems. In *50 Years of Integer Programming*, pages 219–241. Springer, 2010.

[16] A. Lingas. Subexponential-time algorithms for minimum weight triangulations and related problems. In *Proceedings of the 10th Canadian Conference on Computational Geometry (CCCG)*, 1998.

[17] D. Marx and A. Sidiropoulos. The limited blessing of low dimensionality: when $1 - 1/d$ is the best possible exponent for d-dimensional geometric problems. In *SoCG*, pages 67–76. ACM, 2014.

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

[19] W. Mulzer and G. Rote. Minimum weight triangulation is NP-hard. In *SoCG*, pages 1–10, 2006.

[20] C. H. Papadimitriou. The Euclidean traveling salesman problem is NP-complete. *Theor. Comput. Sci.*, 4(3):237–244, 1977.

[21] J. Qian and C. A. Wang. How much precision is needed to compare two sums of square roots of integers? *Inf. Process. Lett.*, 100(5):194–198, 2006.

[22] S. Rao and W. D. Smith. Approximating geometrical graphs via ”spanners” and ”banyans”. In *STOC*, pages 540–550. ACM, 1998.

[23] W. D. Smith and N. C. Wormald. Geometric separator theorems & applications. In *FOCS*, pages 232–243. IEEE Computer Society, 1998.

[24] G. J. Woeginger. Open problems around exact algorithms. *Discrete Applied Mathematics*, 156(3):397–405, 2008.