TSP in a Simple Polygon

Henk Alkema
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Mark de Berg
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Morteza Monemizadeh
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Leonidas Theocharous
Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands

Abstract

We study the Traveling Salesman Problem inside a simple polygon. In this problem, which we call TSP IN A SIMPLE POLYGON, we wish to compute a shortest tour that visits a given set S of n sites inside a simple polygon P with m edges while staying inside the polygon. This natural problem has, to the best of our knowledge, not been studied so far from a theoretical perspective. It can be solved exactly in $\text{poly}(n, m) + 2^{O(\sqrt{n}\log n)}$ time, using an algorithm by Marx, Pilipczuk, and Pilipczuk (FOCS 2018) for SUBSET TSP as a subroutine. We present a much simpler algorithm that solves TSP IN A SIMPLE POLYGON directly and that has the same running time.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Traveling Salesman Problem, Subexponential algorithms, TSP with obstacles

Digital Object Identifier 10.4230/LIPIcs.ESA.2022.5

Funding HA, MdB, and LT are supported by the Dutch Research Council (NWO) through Gravitation-grant NETWORKS-024.002.003.

1 Introduction

The TRAVELING SALESMAN PROBLEM, or TSP for short, is a classic algorithmic problem. Given an edge-weighted graph, the goal is to compute a *tour* – a cycle that visits each node exactly once – of minimum total weight. The problem has been studied widely; in fact, TSP is probably one of the most intensely studied problems in optimization and computer science. There are even several books devoted to TSP [2, 9, 15, 29]. Important special cases of TSP are METRIC TSP and EUCLIDEAN TSP. In METRIC TSP, the edge weights in the input graph form a metric and, in particular, satisfy the triangle inequality. The algorithm by Christofides [8] gives a (3/2)-approximation for this version of the problem. A special case of METRIC TSP is EUCLIDEAN TSP. Here the input is a set S of n points in $\mathbb{R}^d$ and the goal is to visit all points with a tour of minimum total Euclidean length. Due to its natural setting, EUCLIDEAN TSP is among the most studied versions of TSP. EUCLIDEAN TSP is NP-hard [12, 27] but, unlike METRIC TSP, it admits a PTAS as was shown in the celebrated papers of Arora [3] and (for 2D) Mitchell [26]. A PTAS with a better running time was later presented by Rao and Smith [28]. Recently, Kisfaludi-Bak *et al.* [20] presented a PTAS with $2^{O(1/\varepsilon^{d-1})} n \log n$ running time, which they proved to be optimal under Gap-ETH. There are also PTASs for TSP in planar graphs [4, 21] and in spaces of constant doubling dimension [5] – as Trevisan [33] proved, the restriction to bounded dimension is necessary for a PTAS, even in Euclidean spaces – and in spaces of so-called bounded global growth [7].

Our focus is on exact algorithms for TSP. The general TSP problem can be solved exactly in $O(2^n n^2)$ time using dynamic programming, as was shown independently by Held and Karp [16] and Bellman [6]. There is no subexponential algorithm – that is, no algorithm with $2^{o(n)}$ running time – for the general problem, under the Exponential-Time Hypothesis (ETH) [10, Theorem 14.6]. This lower bound even holds for METRIC TSP. On the other hand, EUCLIDEAN TSP can be solved in subexponential time. Already in the early 1990s, Kann [18] and Hwang, Chang and Lee [17] gave $2^{O(\sqrt{n}\log n)}$ algorithms for the planar version of the problem.[1] This was generalized by Smith and Wormald [31], who presented an $2^{O(n^{1-1/d}\log n)}$ algorithm for EUCLIDEAN TSP in $\mathbb{R}^d$; here and in the sequel we consider the dimension d to be a fixed constant. For a long time it was open if EUCLIDEAN TSP in $\mathbb{R}^d$ admits an exact algorithm with a running time $2^{O(n^{1-1/d})}$. Recently, the question was settled by De Berg *et al.* [11], who presented such an algorithm and showed that no $2^{o(n^{1-1/d})}$ algorithm exists unless ETH fails. There is also an exact algorithm for TSP in hyperbolic spaces of Gaussian curvature -1, which runs in quasi-polynomial time if the minimum distance between any two points is at least some fixed constant $\alpha > 0$ [19]. Finally, Klein and Marx [22] presented an algorithm for SUBSET TSP on planar graphs with integer edge weights, where the goal is to computes a shortest tour visiting a given subset of the vertices. Their algorithm runs in time $\text{poly}(|V|) \cdot 2^{O(\sqrt{n}\log n)} + W)$, where $|V|$ is the total number of vertices, n is the number of vertices to be visited, and W is the maximum edge weight. This was later improved by Marx, Pilipczuk, and Piliczuk [24] who presented an algorithm with running time $\text{poly}(|V|) \cdot 2^{O(\sqrt{n}\log n)}$ that does not need the weights to be bounded or integral.

A natural generalization of EUCLIDEAN TSP is to consider a salesman who is moving among a set of obstacles in the plane, or some higher-dimensional space. We call this problem TSP WITH OBSTACLES. As far as we know, and to our surprise, TSP WITH OBSTACLES seems not to have been studied at all from a theoretical perspective, although it has appeared in a behavioural study where subjects were tested on finding the optimal tour [30]. (There is also a paper describing a genetic algorithm for TSP in the presence of obstacles [13], although the setting is slightly different.) A somewhat related problem is where one is given two simple disjoint polygons, and the salesman must visit all vertices of the polygons without crossing the boundary of the polygons. This problem is markedly different from TSP WITH OBSTACLES, however. In particular, it is not a generalization of EUCLIDEAN TSP, and it has been shown by Abrahamson and Shokoufandeh [1] to be solvable in polynomial time.

The fact that TSP WITH OBSTACLES has not been studied is remarkable, since motion-planning and shortest-path problems are among the most widely studied problems in computational geometry. It is beyond the scope of our paper to give an overview of this area, so we just mention one result that we will need. Guibas and Hershberger [14] show how to construct, for a simple polygon P with m vertices and a source point s, a shortest-path map in $O(m)$ time. The shortest-path map allows us to compute, for a point set S of n points in P and a given source point $s \in S$, the shortest-path tree rooted at s and going to all other points in S, in $O(m + n\log m)$ time. If we do this for all points from S as source point, we obtain all shortest paths (and all distances) between the points in S in $O(nm + n^2\log n)$ time.

TSP WITH OBSTACLES can be solved by computing a shortest path between each pair of points in S, and then running the Bellman-Held-Karp algorithm [6, 16] using the computed pairwise distances, but this leads to an exponential running time. In the plane a much faster algorithm can be obtained if one uses an algorithm for SUBSET TSP as a subroutine, as follows. First, compute all pairwise shortest paths. Next, turn this collection of paths

[1] Comparing the exact lengths of two given tours in the plane is in fact non-trivial, as it involves comparing sums of square roots. To focus on the combinatorial complexity of the problem, such algebraic issues are typically ignored by working in the real-RAM model, which we do as well.

in the plane into a weighted planar graph by inserting a vertex at every intersection point between two paths, where the weight of an edge in the graph is the shortest-path distance between its endpoints. Note that some pairs of path may overlap instead of intersect; in that case, insert a vertex at the two outermost points where they overlap. The resulting graph has $O(n^4)$ vertices – the points in S plus the intersection points – of which n nodes must be visited. Solving SUBSET TSP using the algorithm of Marx, Pilipczuk, and Piliczuk [24] leads to an algorithm with $\text{poly}(n,m) + O(n^2) \cdot 2^{O(\sqrt{n} \log n)} = \text{poly}(n,m) + 2^{O(\sqrt{n} \log n)}$ running time, where m is the total number of edges of the obstacles.

Our contribution. We are interested in the variant of TSP WITH OBSTACLES where the salesman is moving inside a simple polygon P with m edges in total, which contains the set S of points to be visited. In other words, there is a single obstacle which is the region outside the polygon. We call this variant TSP IN A SIMPLE POLYGON. From now on, we refer to the points in S as *sites*, to distinguish them from arbitrary points in P.

As mentioned, TSP WITH OBSTACLES in the plane can be solved in subexponential time using the algorithm of Marx, Pilipczuk, and Piliczuk for SUBSET TSP as a subroutine. Hence, TSP IN A SIMPLE POLYGON can be solved in $\text{poly}(n,m)+2^{O(\sqrt{n} \log n)}$ time. Unfortunately, the algorithm of Marx, Pilipczuk, and Piliczuk for SUBSET TSP is complicated: the description of the algorithm and its correctness proof take 27 pages in total [23]. Here we only count the overview of the algorithm (Section 2, comprising 7 pages) and the detailed description of the algorithm and proof of correctness (Section 5, comprising 20 pages), and not the description of some of the tools being used (Sections 3 and 4, comprising 2.5 pages).

We present a much simpler algorithm with the same running time, based on the elegant algorithm by Hwang, Chang and Lee [17] for EUCLIDEAN TSP in the plane. We also prove several basic properties of optimal solutions for TSP WITH OBSTACLES, which are of independent interest.

2 Notation and basic properties

In this section we introduce the notation and terminology used throughout paper, and we prove several basic properties of optimal TSP tours in a simple polygon.

Let P be a simple polygon, which is the region in which the salesman can move. We consider P to be a closed set, so that shortest paths are well defined. Let $S = \{s_1, \ldots, s_n\} \subset P$ denote the set of sites to be visited by the salesman. We say that a vertex v of P is *reflex* if the angle between the two edges incident to v, measured inside P, is more than 180 degrees.

Whenever we speak of a *path*, we mean a path that stays in P, unless stated otherwise. We denote the length of the line segment pq connecting points p and q by $|pq|$. Similarly, we denote the (Euclidean) length of a path π by $|\pi|$. For two points $p, q \in P$ we use $\pi(p,q)$ to denote the (unique) shortest path between them, that is, the path from p to q that has minimum length while staying inside P. Finally, for a path π and two points $a, b \in \pi$, the subpath of π between a and b is denoted by $\pi[a,b]$.

Consider a polygonal path π (that is, a path consisting of straight-line segments) and let v be a reflex vertex of P. We say that π *bends around* v if v coincides with an interior vertex (not an endpoint) of π and π is *locally shortest* at v. In other words, if e_1, e_2 are the two edges of π meeting at v, then the two edges of P meeting at v lie in the convex wedge defined by e_1 and e_2. Observe that a shortest path π inside P must bend around a reflex vertex of P at each of the path's interior vertices. The next lemma shows that in a simple polygon, local optimality implies global optimality. It is folklore, but for completeness we give a proof.

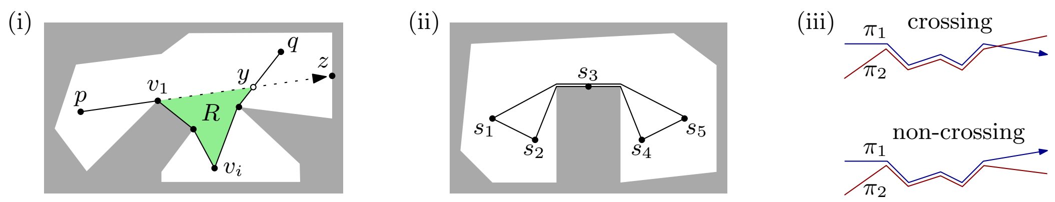

Figure 1 (i) Illustration for the proof of Lemma 1. (ii) An optimal tour may pass through the same site twice, and can contain (partially) overlapping line segments. (iii) Crossing and non-crossing paths. Note: paths in parts (ii) and (iii) are shown slightly displaced where they overlap.

▶ **Lemma 1.** *Let p, q be two points inside a simple polygon P and let π be a path between p and q such that every interior vertex of π bends around a reflex vertex of P. Then, π is the shortest path from p to q in P.*

Proof. Let $v_1, \ldots, v_k$ be the interior vertices of π, so $\pi = (p, v_1, ..., v_k, q)$. We will prove the lemma by induction on k. For $k = 0$ the lemma trivially holds, so assume $k > 0$.

Consider the ray from p through v_1, and let z be the first point where the ray hits ∂P after v_1. Then the segment v_1z splits P into two parts. Let P_1 be the part that contains p, and let P_2 be the other part. We claim that $q \in P_2$. Indeed, if this was not the case then π must cross v_1z at some point y. Consider the region $R \subset P$ enclosed by v_1y and $\pi[v_1, y]$; see Fig. 1(i). (Our assumptions do not immediately rule out that π intersects itself, so R need not be simple. But this does not invalidate the coming argument.) Since P is simple, R cannot contain any part of ∂P. Let v_i denote a vertex of π on $\pi[v_1, y]$ that has maximal distance from v_1y. Then the angle between the two edges of π incident to v_i has to be convex when measured inside R, otherwise v_i cannot be a vertex of maximal distance from v_1y. By assumption, π has to bend around a vertex of P at v_i, but this cannot happen since $R \subset P$. Hence, $q \in P_2$, as claimed.

We now observe that the shortest path from p to any point in P_2 passes through v_1. Moreover, by the induction hypothesis we know that the path $(v_1, ..., v_k, q)$ is the shortest path from v_1 to q. We conclude that π is the shortest path from p to q in P. ◀

It is well known that optimal tours for EUCLIDEAN TSP in $\mathbb{R}^2$ are non-self-intersecting, that is, if s_is_j and s_ks_t are non-consecutive edges in an optimal tour then s_is_j and s_ks_t do not intersect (except when all sites are collinear). This is no longer true for TSP IN A SIMPLE POLYGON: two paths $\pi(s_i, s_j)$ and $\pi(s_k, s_t)$ that are part of an optimal tour can meet in one or more points, and sites may be visited more than once; see Fig. 1(ii). However, we can still formulate a non-crossing property for TSP WITH OBSTACLES, as shown next.

Let $\pi_1 = \pi(p_1, q_1)$ and $\pi_2 = \pi(p_2, q_2)$ be two shortest paths in P. Observe that their intersection $\pi_1 \cap \pi_2$, may contain at most one connected component, due to the uniqueness of shortest paths. Let γ denote this component. We give an arbitrary orientation to the path π_1. We say that π_1 and π_2 are *crossing* when π_2 lies on opposite sides of π_1 just before γ and just after γ, relative to the chosen orientation of π_1; otherwise π_1 and π_2 are *non-crossing*. See Fig. 1(iii) for an example.

We now state the main result of this section.

▶ **Theorem 2.** *Let P be a simple polygon and let S be a set of n sites in P. Then there is an optimal tour T_{opt} through S such that*

1. *T_{opt} passes at most twice through any point of P, and*
2. *any two paths of T_{opt} are non-crossing.*

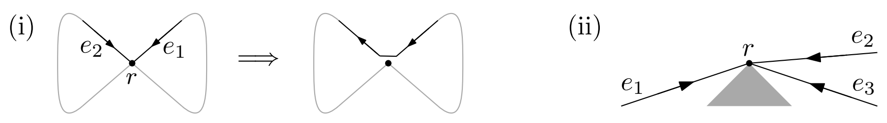

Figure 2 Illustrations for the proof of Theorem 2.

▶ Remark 3. A site s_i may lie on the shortest path between two other sites s_j, s_k, in which case T_{opt} may use $\pi(s_j, s_k)$ plus two shortest paths with s_i as endpoint; see site s_3 in Fig. 1(ii). This does not contradict Property 1 of the theorem: T_{opt} still passes through s_i only twice.

Proof.

Proof of part 1. We first observe the following. Let r be a point through which T_{opt} passes at least twice. Give T_{opt} an arbitrary orientation, and suppose T_{opt} first arrives at r along a segment e_1 and later along a segment e_2 that is not collinear with e_1. Then r must be a reflex vertex of P, and P must, locally at r, lie in the convex wedge defined by e_1 and e_2; otherwise we could shortcut T_{opt} at r as shown in Fig. 2(i).

Now suppose T_{opt} passes at least three times through some point r. Then there are at least three segments e_1, e_2, e_3 through which T_{opt} arrives at r. If two of these segments overlap then we can shortcut T_{opt}, which is a contradiction. Otherwise at least two of the three wedges defined by e_1, e_2, e_3 have an opening angle less than 180 degrees. The point r can be a reflex vertex of R, but locally at r, the polygon P can lie in only one of these two wedges; see Fig. 2(ii). Hence, we can shortcut at the other wedge, which is a contradiction.

Proof of part 2. Number the sites in order along T_{opt} so that $T_{\text{opt}} = \bigcup_{\ell=1}^{n} \pi_\ell$, where we define $\pi_\ell := \pi(s_\ell, s_{\ell+1})$ and $s_{n+1} := s_1$. Suppose there are two paths π_i and π_j that cross each other. Notice that the paths $\pi(s_i, s_j)$ and $\pi(s_{i+1}, s_{j+1})$ do not cross each other; see Fig. 1, where the two crossing paths shown at the top can be "uncrossed", resulting in the non-crossing paths shown at the bottom. Let Γ_1 be the part of T_{opt} from s_{i+1} to s_j and let Γ_2 be the part of T_{opt} from s_{j+1} to s_i. Then $T^*_{\text{opt}} := \Gamma_1 \cup \pi(s_{i+1}, s_{j+1}) \cup \Gamma_2 \cup \pi(s_i, s_j)$, where the direction of Γ_1 is reversed, is a tour of the same length as T_{opt}. Moreover, T^*_{opt} has fewer crossings; this is true because uncrossing π_i and π_j cannot generate new crossings by part 1 of the theorem. Hence, we can repeat the process until we are left with an optimal tour without any crossings. ◀

3 A subexponential algorithm for TSP IN A SIMPLE POLYGON

In this section we give a subexponential algorithm for TSP IN A SIMPLE POLYGON, based on the work by Hwang *et al.* [17]. The algorithm of Hwang *et al.* solves a problem that is slightly more general than TSP, so that it can be used in a recursive algorithm. In a recent paper, De Berg *et al.* [11] define the same generalized problem, but with a different terminology that will be more convenient for us. In what follows we will mostly use their terminology to define the problem, which De Berg *et al.* call EUCLIDEAN PATH COVER. (Hwang *et al.* called it GENERALISED EUCLIDEAN TSP).

Let $S' \subset S$ be a subset of n of the sites of the initial set of sites, let $B \subset S'$ be a set of *boundary sites*, and let M be a perfect matching on B. We say that a collection of paths[2] $\mathcal{P} = \{\pi_1, \pi_2, ..., \pi_{|B|/2}\}$ *realizes* M *on* S' if (i) for each pair $(s_i, s_j) \in M$ there is a path

[2] Note that we are now temporarily back in the standard EUCLIDEAN TSP setting. Thus the paths mentioned here are not shortest paths between sites, but paths in the complete graph on the sites. In the current setting, the edges of these paths are straight-line segments between the corresponding sites; when we go back to TSP IN A SIMPLE POLYGON, they will be shortest paths between sites.

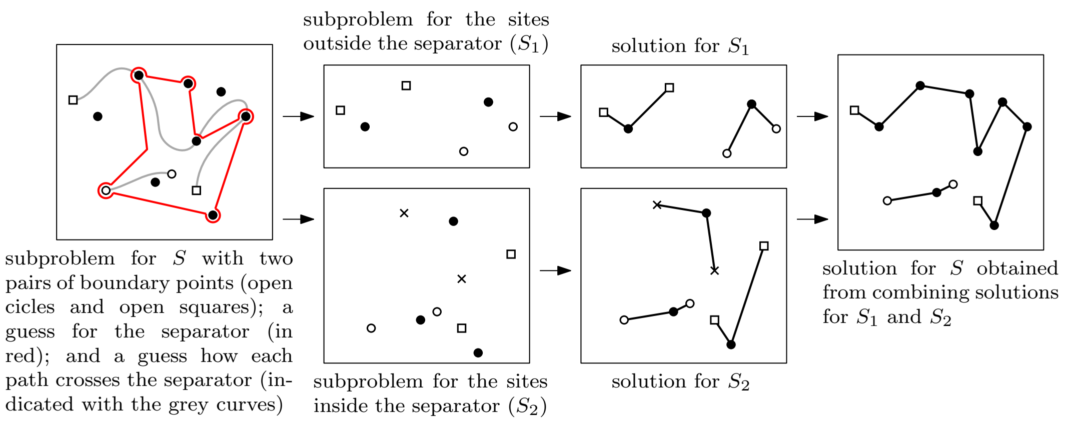
subproblem for the sites outside the separator (S_1)
solution for S_1
subproblem for S with two pairs of boundary points (open cicles and open squares); a guess for the separator (in red); and a guess how each path crosses the separator (indicated with the grey curves)
subproblem for the sites inside the separator (S_2)
solution for S_2
solution for S obtained from combining solutions for S_1 and S_2

Figure 3 An example of how the algorithm works. After trying all possible separators and all possible ways the paths cross the separator, we will have found the optimal solution.

$\pi_{ij} \in \mathcal{P}$ with s_i and s_j as endpoints, and (ii) the paths together visit each site in S' exactly once. The goal of EUCLIDEAN PATH COVER is to find a collection of paths of minimum total length that realizes M on S'. Note that we can solve EUCLIDEAN TSP by solving $n-1$ instances of EUCLIDEAN PATH COVER on S', namely for $B = \{s_1, s_i\}$ where $i \in \{2, \ldots, n\}$, and taking the best solution found. Next we describe how an instance of EUCLIDEAN PATH COVER can be solved using the triangulation approach. For simplicity, and with a slight abuse of notation, we will denote the set of sites that need to be visited by S (rather than S').

The triangulation approach. Hwang *et al.* solve EUCLIDEAN PATH COVER with a separator-based divide-and-conquer algorithm, which we will refer to as the *triangulation approach.* It works as follows. We start by creating Q, a set of three points defining an arbitrarily large triangle containing all the points in S. Suppose we have a maximal triangulation $\mathcal{T}$ of $S \cup Q$ that uses all segments from each path in an optimal solution $\mathcal{P}_{\text{opt}}$. Since $\mathcal{T}$ is a maximal planar graph, Miller's separator theorem [25] implies that there exists a simple cycle $\mathcal{C}$ in $\mathcal{T}$ of at most $2\sqrt{2(n+3)}$ points from $S \cup Q$ such that at most $2(n+3)/3$ sites are inside $\mathcal{C}$, and at most $2(n+3)/3$ sites are outside $\mathcal{C}$. The idea is to use the separator $\mathcal{C}$ to split the problem into two subproblems: one inside $\mathcal{C}$ and one outside $\mathcal{C}$, and glue the solutions to these subproblems together to get the solution to the original problem. However, we have to make sure that the solutions inside and outside match with each other. To ensure this, we guess all possible ways in which $\mathcal{P}_{\text{opt}}$ can cross $\mathcal{C}$, and handle each of them separately. Since the triangulation uses all segments from each path in $\mathcal{P}_{\text{opt}}$, no edge in $\mathcal{P}_{\text{opt}}$ crosses an edge in $\mathcal{C}$. Hence, $\mathcal{P}_{\text{opt}}$ only goes from inside $\mathcal{C}$ to outside $\mathcal{C}$ via nodes of $\mathcal{C}$. Thus guessing where $\mathcal{P}_{\text{opt}}$ crosses $\mathcal{C}$ amounts to guessing for each path in $\mathcal{P}_{\text{opt}}$ at which nodes of $\mathcal{C}$ it crosses $\mathcal{C}$ (as well as the order of these crossing nodes). Each guess will lead to different pairs of subproblems to be solved inside and outside the separator. The optimal solution will then be the best solution over all guesses. See Fig. 3 for an example.

Above we assumed that we had a triangulation $\mathcal{T}$ of $S \cup Q$ that uses all segments from each path in $\mathcal{P}_{\text{opt}}$. But we do not know $\mathcal{P}_{\text{opt}}$, since $\mathcal{P}_{\text{opt}}$ is what we want to compute. We therefore try all possible simple cycles $\mathcal{C}$ of at most $2\sqrt{2(n+3)}$ sites, and for each of them compute an optimal solution under the assumption that the solution does not use any edge that intersects an edge in the cycle. One of these guesses must correspond to the separator for the (unknown) triangulation $\mathcal{T}$. Algorithm 1 gives a high-level description of the algorithm. Next we show how to apply the algorithm to TSP IN A SIMPLE POLYGON, and fill in the details for the various steps.

Algorithm 1 *Triangulation-Approach*(S, B, M).

```
Input: set $S$ of $n$ sites; set $B \subseteq S$ of boundary points; perfect matching $M$ on $B$
Output: A solution for EUCLIDEAN PATH COVER for the instance $(S, B, M)$
1: if $|S|$ is a sufficiently small constant then
2:     Compute an optimal solution $\mathcal{P}_{\text{opt}}$ by brute-force
3: else
4:     Generate all candidate separators $\mathcal{C}$ for $S$, as explained in the text
5:     for each candidate separator $\mathcal{C}$ do
6:         Generate all pairs of subproblems $(S_1, B_1, M_1), (S_2, B_2, M_2)$ for $\mathcal{C}$,
           as explained in the text. Recursively solve each pair of subproblems, and let
           $\mathcal{P}_{\text{opt}}(S_1, B_1, M_1, S_2, B_2, M_2)$ be the solution obtained by concatenating
           the solutions to the subproblems.
7: $\mathcal{P}_{\text{opt}} \leftarrow$ best of all solutions $\mathcal{P}_{\text{opt}}(S_1, B_1, M_1, S_2, B_2, M_2)$ from the previous step
8: return $\mathcal{P}_{\text{opt}}$
```

3.1 Applying the triangulation approach to TSP IN A SIMPLE POLYGON

We now return to TSP IN A SIMPLE POLYGON. As above, we will solve a path-cover problem, which we call GENERALIZED EUCLIDEAN PATH COVER. This problem is the same as EUCLIDEAN PATH COVER, except that the edges used by the paths, which used to be straight-line segment connecting sites, are now going to be shortest paths connecting sites.

To apply the triangulation approach in our new setting, we need a maximal triangulation of an optimal solution $\mathcal{P}_{\text{opt}}$ to the given instance of the GENERALIZED EUCLIDEAN PATH COVER. The existence of such a maximal triangulation implies that there is a small separator by Miller's separator theorem, which we can then guess. There exists of course a triangulation of $\mathcal{P}_{\text{opt}}$ if we are allowed to use all internal vertices of these paths as vertices in the triangulation: we just need to construct a so-called *constrained triangulation* on the set of all sites and internal shortest-path vertices, which uses the given segments on the paths. But the number of nodes of such a triangulation would not only depend on the number of sites, but also on the total complexity of the shortest paths, which depends on the complexity of the polygon P. The key idea to overcome this, is to work with a triangulation whose nodes are the sites in S and whose edges are shortest paths. Next, we show that such a triangulation always exists, and show how to create a maximal triangulation from this triangulation. After that we will describe the various steps of the algorithm in more detail, and we will prove its correctness and analyze its running time.

Triangulating $\mathcal{P}_{\text{opt}}$. Recall that a set $X \subseteq P$ is called *geodesically convex* if for any two points $p, q \in X$ the shortest path $\pi(p, q)$ in P is contained in X. The *relative convex hull* [32] of a set S of sites inside a simple polygon P, denoted by RCH(S), is defined as the intersection of all geodesically convex sets containing S. Intuitively, RCH(S) is obtained by placing a rubber band on ∂P, and then releasing the band so that it "snaps" around S, without crossing over any site and while staying inside P. The boundary of RCH(S) consists of shortest paths connecting points of S; see Fig. 4(i) for an illustration.

Let $E(S) := \{\pi(s_i, s_j) : s_i, s_j \in S \text{ and } i \neq j\}$ to be the set of all shortest paths between the sites in S. We now define an *sp-triangulation* (see Figure 4(ii)) of S to be a collection $\mathcal{T}_{\text{sp}}(S) \subset E(S)$ of pairwise non-crossing shortest paths between the sites in S such that

- $\mathcal{T}_{\text{sp}}(S)$ includes the shortest paths that form ∂RCH, and
- each face in the subdivision of RCH(S) defined by $\mathcal{T}_{\text{sp}}(S)$ (after moving pieces of the paths slightly apart where they overlap) is bounded by exactly three paths.

▶ **Lemma 4.** *For any set S of sites inside a simple polygon P, and any given collection $E^* \subset E(S)$ of pairwise non-crossing paths, there exists an sp-triangulation $\mathcal{T}_{\text{sp}}(S)$ that includes the paths from E^*.*

Proof. Consider the subdivision of RCH(S) induced by (the boundary of RCH(S) and) the prescribed shortest paths in E^*. We will show how to triangulate each face F in this subdivision using paths from $E(S)$.

The face F has an outer boundary and it contains zero or more additional shortest paths. Let $E_{\text{out}}(F)$ denote the shortest paths forming the outer boundary, let $E_{\text{in}}(F)$ denote the remaining shortest paths, and let $E(F) := E_{\text{out}}(F) \cup E_{\text{in}}(F)$. Let $S_{\text{out}}(F)$ denote the sites from S on the outer boundary of F, let $S_{\text{in}}(F)$ denote the remaining sites, and let $S(F) = S_{\text{out}}(F) \cup S_{\text{in}}(F)$. If $S_{\text{in}}(F) = \emptyset$ and $|S_{\text{out}}(F)| = 3$ then F is already a triangle and we are done. Otherwise we will argue that there exist two sites $s_i, s_j \in S(F)$ such that

- the shortest path $\pi(s_i, s_j)$ lies inside F
- $\pi(s_i, s_j)$ is not yet present in $E(F)$ and
- $\pi(s_i, s_j)$ does not cross any shortest path in $E(F)$

These properties imply that we can add $\pi(s_i, s_j)$ to $E(F)$. This either splits F into two faces, or it adds a path to $E_{\text{in}}(F)$. In both cases we can continue recursively on the resulting face(s), until every face is a triangle. It remains to argue the existence of the pair s_i, s_j. We consider two cases:

Case I: $S_{\text{in}}(F) = \emptyset$ and all sites from $S_{\text{out}}(F)$ are convex vertices of F.
Then take two sites $s_i, s_j \in S(F)$ that are not already connected by a shortest path in $E_{\text{out}}(F)$. Such a pair exists because $|S_{\text{out}}(F)| > 3$. Consider the shortest path π from s_i to s_j *that is restricted to lie inside* F. All interior vertices of π must bend around reflex vertices of F. Since all vertices from $S_{\text{out}}(F)$ are convex, these reflex vertices are not sites but reflex vertices of P. By Lemma 1, π must also be the shortest path in P between p and q and so $\pi \in E(S)$. Since π stays inside F, it does not cross any other shortest path.

Case II: $S_{\text{in}}(F) \neq \emptyset$ or $S_{\text{out}}(F)$ contains a site that is a reflex vertex of F.
We start by identifying a "reflex" site, from which we will then be able to add a path.

▷ Claim. There is a site $s_i \in S(F)$ with the following property: there is a line ℓ through s_i such that all paths from $E(F)$ that have s_i as an endpoint lie to the same side of ℓ, locally near s_i. (In other words, the edges of these paths that have s_i as an endpoint all lie to the same side of ℓ).

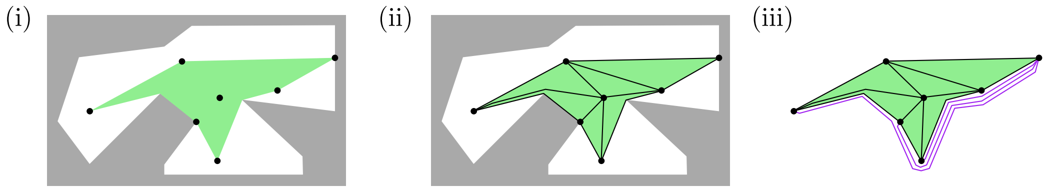

Figure 4 (i) The relative convex hull of a set of points. (ii) An sp-triangulation. (iii) The corresponding maximal triangulation. Note: paths are shown slightly displaced where they overlap.

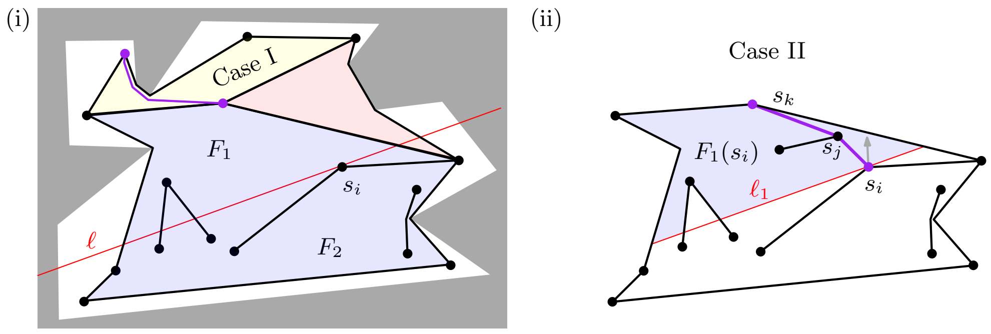

Figure 5 Illustrations for the proof of Lemma 4. (i) An example with three faces inside RCH(S). The red face is already a triangle. The yellow face falls into Case 1 and can be triangulated by adding the purple path. The blue face falls into Case 2 of the proof. (ii) The ray from s_i hits a shortest path that has s_k as an endpoint in $F_1(s_i)$. The purple path is the shortest path from s_i to s_k inside $F_1(s_i)$, and therefore $\pi(s_i, s_j)$ is added to the triangulation.

Proof. First suppose that $S_{\text{in}}(F) = \emptyset$. Then $S_{\text{out}}(F)$ contains a site s_i that is a reflex vertex of F and, hence, has the required property. (Note that, except for the two paths on the outer boundary of F meeting at s_i, the site s_i has no other incident paths, since $S_{\text{in}}(F) = \emptyset$.)

Now suppose $S_{\text{in}}(F) \neq \emptyset$. Consider a connected component C in $E_{\text{in}}(F)$. (Or more precisely, a connected component of the set $\bigcup E_{\text{in}}(F) \subset \mathbb{R}^2$.) Then C may contain at most one site from $S_{\text{out}}(F)$ – indeed, if it would contain two such sites then it would split F and some paths in the component would be part of the outer boundary of F, a contradiction. Now see $E_{\text{out}}(F)$ as a simple polygon and consider the relative convex hull of $C \cap S(F)$ within this simple polygon. Then choose s_i to be any vertex of RCH($C \cap (S(F))$ that is a site from $S_{\text{in}}(F)$. Then there is a line ℓ through s_i such that the path edges incident to s_i lie to the same side of ℓ, namely a line ℓ that is (locally) tangent to RCH($C \cap S(F)$). ◁

Consider the site s_i and the line ℓ provided by the above claim. Let $\ell_1 \subset \ell$ denote the maximal subsegment of ℓ containing s_i and lying inside F. The segment ℓ_1 splits F in two pieces, F_1 and F_2. Let F_2 denote the piece containing the edges incident to s_i and let F_1 denote the other piece. We claim that F_2 must contain all sites s_k already connected to s_i, that is, all sites $s_k \in S(F)$ such that $\pi(s_i, s_k) \in E(F)$. Indeed, if some neighbor s_k of s_i was in F_1 then $\pi(s_i, s_k)$ would intersect ℓ_1 at some point s and therefore we would be able to shorten $\pi(s_i, s_k)$ by using the straight segment $s_i s$ as a shortcut. Now, ℓ_1 together with the paths in $E_{\text{in}}(F)$ induces a subdivision of F into planar regions. Denote by $F_1(s_i)$ the region in this subdivision within F_1 that contains s_i on its boundary; see Fig. 5(ii), where this region is shown in purple.

Now shoot a ray from s_i into $F_1(s_i)$, and let π denote the first path from $E(F)$ hit by this ray. At least one of the two endpoints of π must be in $F_1(s_i)$, otherwise π would intersect ℓ_1 twice, contradicting that π is is a shortest path. Let s_k be this endpoint. Consider the shortest path γ from s_i to s_k, *restricted to lie in* $F_1(s_i)$. Then γ has to bend around vertices of $F_1(s_i)$, which are either reflex vertices of P or sites in $S(F)$. Let s_j be the first vertex on γ that is a site; such a vertex exists, since the endpoint s_k of γ is a site. Then all interior vertices of $\gamma[s_i, s_j]$, the subpath of γ from s_i to s_j, bend around reflex vertices of P. By Lemma 1, the subpath $\gamma[s_i, s_j]$ is a shortest path in P. Hence, we have found a shortest $\pi(s_i, s_j)$ that we can add to $E(F)$. ◀

Note that $\mathcal{T}_{\rm sp}(S)$, when viewed as a graph, may not be maximally planar. The reason is that the face outside $\textsc{rch}(S)$ is not necessarily a triangle. However, we can easily turn $\mathcal{T}_{\rm sp}(S)$ into a maximally planar graph, as follows. Let $s_1, ..., s_r$ be the sites on $\partial\textsc{rch}(S)$, in the same order as they are encountered while following $\partial\textsc{rch}(S)$ clockwise. For every pair (s_1, s_i) with $3 \leqslant i \leqslant r-1$, we add the path from s_1 to s_i that follows $\partial\textsc{rch}$ clockwise, as an edge to our triangulation. We denote this set of "outside edges" by $\overline{E}(S)$; see the purple paths in Figure 4(iii) for an example. By adding these edges to $\mathcal{T}_{\rm sp}(S)$, we triangulate the outer face and thus obtain a maximally planar graph, which we denote by $\mathcal{G}_{\rm sp}(S)$.

Now that we have established the existence of a suitable triangulation, we can explain how to implement the various steps in Algorithm 1.

Preprocessing. As a preprocessing step we compute the shortest path $\pi(s_i, s_j)$ for each pair $s_i, s_j \in S$, which also gives us all pairwise distances. We then check for each pair of shortest paths whether they cross or not, so that later on in the algorithm we have this information readily available. Clearly, the preprocessing can be done in poly(n, m) time. As it turns out, Algorithm 1 will not need the actual shortest paths or other geometric information – knowing pairwise distances (for solving the constant-size subproblems at the base of the recursion) and whether or not pairs of shortest paths cross (for larger subproblems) is all that is needed.

Generating all candidate separators. Miller's theorem [25] guarantees that $\mathcal{G}_{\rm sp}(S)$ has a simple-cycle separator of at most $2\sqrt{2n}$ nodes. The edges in this separator are edges in $\mathcal{G}_{\rm sp}(S)$, which correspond to paths in $E(S) \cup \overline{E}(S)$. Note that for every pair of sites, there are at most two different paths in $E(S) \cup \overline{E}(S)$. We now generate our collection of candidate separators as follows:

```
1: for every circular sequence $s_1, s_2, \ldots, s_t, s_1$ of at most $2\sqrt{2n}$ sites from $S$ do
2:     for every choice of paths from $E(S) \cup \overline{E}(S)$ to connect consecutive sites do
3:         Check if the paths used in the resulting cycle $\mathcal{C}$ are pairwise non-crossing and if
           the number of sites inside and outside $\mathcal{C}$ is at most $2n/3$. If so, add $\mathcal{C}$ to the
           collection of cycles.
```

Clearly, the collection of generated separators is guaranteed to contain the separator that would result from applying Miller's theorem to $\mathcal{G}_{\rm sp}(S)$. The total number of candidate separators is $n^{O(\sqrt{n})}$ (for choosing the circular sequence) times $2^{O(\sqrt{n})}$ (for choosing the paths to connect consecutive sites), so $n^{O(\sqrt{n})} \cdot 2^{O(\sqrt{n})} = 2^{O(\sqrt{n}\log n)}$ in total.

Note that for each candidate separator $\mathcal{C}$ we need to check if it is simple, that is, if its paths are pairwise non-crossing. With the pre-computed information, this can be done in $O(|\mathcal{C}|^2) = O(n)$ time. Note that the paths from $\overline{E}(S)$ do not cross any other path by definition, so these paths need not be checked.

If $\mathcal{C}$ is simple, we need to determine which sites are inside $\mathcal{C}$ and which are outside $\mathcal{C}$. In fact, what is "inside" and what is "outside" is not important, we just need to partition the set of sites (that are not on the separator) into two subsets: the sites on one side of $\mathcal{C}$ and the sites on the other side of $\mathcal{C}$. We can do that as follows.

Take any site s_i that is not on the separator $\mathcal{C}$, and consider another site s_j (that is not on $\mathcal{C}$ either). We can decide if s_j is on the same side of $\mathcal{C}$ as s_i by counting how often $\pi(s_i, s_j)$ crosses $\mathcal{C}$: site s_j is on the same side if and only if $\pi(s_i, s_j)$ crosses $\mathcal{C}$ an even number of times. Because we determined in the preprocessing step for each pair of shortest paths whether they cross or not, we can do this counting in $O(\sqrt{n})$ time in total. The separator may also use paths from $\overline{E}(S)$ and these paths depend on the subproblem being solved. Hence, they have not been considered in the preprocessing step. However, such paths cannot cross $\pi(s_i, s_j)$ by definition, so they can be ignored. This also means that we do not have to compute the paths in $\overline{E}(S)$ explicitly. We conclude that, for each candidate separator $\mathcal{C}$, we can partition the sites not on $\mathcal{C}$ into subsets on either side of the separator in $O(n\sqrt{n})$ time.

Hence, in total we spend $O(n\sqrt{n}) \cdot 2^{O(\sqrt{n}\log n)} = 2^{O(\sqrt{n}\log n)}$ time to generate all possible separators and their corresponding partitions.

▶ **Lemma 5.** *The number of separators generated in Step 4 of Algorithm 1 is $2^{O(\sqrt{n}\log n)}$, and they can be generated in the same amount of time. This includes determining for each candidate $\mathcal{C}$ a partitioning of $S \setminus \mathcal{C}$ into sites on one side of $\mathcal{C}$ and on the other side of $\mathcal{C}$.*

Generating the subproblems for a given separator $\mathcal{C}$. This can be done as in the algorithms of Hwang *et al.* [17] and De Berg *et al.* [11]; see Fig. 3. For completeness we give a sketch.

Consider the set $\mathcal{C} \setminus B$, which contains the sites from $\mathcal{C}$ that are not boundary sites. The path $\xi_{ij} \in \mathcal{P}_{\text{opt}}$ corresponding to a pair $(s_i, s_j) \in M$ may or may not visit sites of $\mathcal{C} \setminus B$. To generate our subproblems, we need to guess for every pair $(s_i, s_j) \in M$, which points of $\mathcal{C} \setminus B$ are visited by ξ_{ij} and in which order. (A site from $\mathcal{C} \setminus B$ should be used by at most one path ξ_{ij}.) Given such a guess, we can generate the corresponding subproblems, one for each side, as follows. Let $s_{k_1}, s_{k_2}, \ldots, s_{k_t}$ be the ordered sequence of sites from $\mathcal{C} \setminus B$ that is our guess for where ξ_{ij} crosses C. Define $\xi'_{ij} := s_i, s_{k_1}, s_{k_2}, \ldots, s_{k_t}, s_j$. Then every pair of consecutive sites of ξ'_{ij} becomes a pair in the matching of one of the subproblems; which subproblem depends on which side we have guessed the corresponding part of ξ_{ij} to be on. After doing this for all ξ_{ij} corresponding to a pair in M, we have the matchings M_1 and M_2 of our subproblems. The sets of sites S_1 and S_2 for the recursive calls are generated as follows. First we add the sites inside $\mathcal{C}$ to S_1 and the sites outside $\mathcal{C}$ to S_2. Then we add the sites of M_1 to S_1, and the sites of M_2 to S_2. Note that causes some sites to appear in both subproblems. Finally, the remaining sites on C are all added to S_1. In other words, we consider all points on C to be on the same side of C. See Fig. 3 for an example.

The total number of subproblems generated for a given separator $\mathcal{C}$ is $n^{O(\sqrt{n})} = 2^{O(\sqrt{n}\log n)}$, since we have to guess for each separator node by which (if any) of the paths it is used and then choose an ordering for the crossings.

A proof of correctness. Next we prove that this gives an optimal solution.

▶ **Lemma 6.** *Let s_i be such that there is an optimal solution T_{opt} with non-crossing shortest paths that uses the path $\pi(s_1, s_i)$. Then Triangulation-approach(S, B, M) with $B = \{s_1, s_i\}$ computes an optimal solution for* TSP IN A SIMPLE POLYGON. *Moreover, all other calls report valid solutions.*

Proof. We first argue that each reported solution is valid, by showing that any recursive call gives a valid solution of EUCLIDEAN PATH COVER. This is trivially true for the base case in the algorithm. The way in which the subproblems are generated and their solutions are combined, ensures that when the solutions to the subproblems are valid (which we can assume by induction) then the combined solution is valid. (Note: the generation of the subproblems and how they are combined is based on the original triangulation approach by Hwang *et al.* [17], so this part in fact follows from the correctness of their algorithm.)

Now consider a call *Triangulation-approach*(S, B, M). Let $\mathcal{P}_{\text{opt}}(S, B, M)$ be an optimal solution to the subproblem. We will prove that if the parameters S, B, M are consistent with T_{opt} – that is, $\mathcal{P}_{\text{opt}}(S, B, M)$ is a subset of the global T_{opt} – then the algorithm computes an optimal solution to the subproblem. Since the initial call with $B = \{s_1, s_i\}$ is consistent with T_{opt} by definition, this will prove the lemma. Note that if $\mathcal{P}_{\text{opt}}(S, B, M) \subset T_{\text{opt}}$, then $\mathcal{P}_{\text{opt}}(S, B, M)$ consists of non-crossing paths. Lemma 4 then implies that there exists an sp-triangulation $\mathcal{T}_{\text{sp}}(S)$ that includes the edges from $\mathcal{P}_{\text{opt}}(S, B, M)$. This can be extended to a maximally planar graph $\mathcal{G}_{\text{sp}}(S)$, which has separator of size at most $2\sqrt{2n}$. As argued earlier, this separator will be one of the generated candidate separators, and the way in which $\mathcal{P}_{\text{opt}}(S, B, M)$ crosses it will be corresponds to one of the generated subproblems. The parameters $S_{\text{in}}, B_{\text{in}}, M_{\text{in}}$ and $S_{\text{out}}, B_{\text{out}}, M_{\text{out}}$ of these subproblems are thus consistent with T_{opt}, and so we can assume by induction that they are solved optimally, thus leading to an optimal solution for the call to *Triangulation-approach*(S, B, M). ◀

Putting it all together. Lemma 6 gives the correctness of our algorithm so it remains to analyze the running time. The preprocessing takes poly(n, m) time. The running time of *Triangulation-approach* satisfies the recurrence $T(n) = 2^{O(\sqrt{n}\log n)} + 2^{O(\sqrt{n}\log n)} \cdot T(\frac{2n}{3} + O(\sqrt{n}))$, which solves to $T(n) = 2^{O(\sqrt{n}\log n)}$. This leads to our final theorem.

▶ **Theorem 7.** TSP IN A SIMPLE POLYGON *for a set S of n sites in a polygon P with m edges can be solved in poly$(n, m) + 2^{O(\sqrt{n}\log n)}$ time.*

4 Conclusion

We introduced TSP IN A SIMPLE POLYGON, a natural variant of TSP that seems not to have been studied at all so far. The problem can be solved in poly$(n, m) + 2^{O(\sqrt{n}\log n)}$ time using a complicated algorithm of Marx, Pilipczuk, and Pilipczuk [24] as a subroutine. We presented a much simpler algorithm with the same running time. Our work raises several questions:

- It was recently shown that EUCLIDEAN TSP in $\mathbb{R}^2$ can be solved in $2^{O(\sqrt{n})}$ time [11]. Can we also get rid of the log-factor in the exponent for TSP IN A SIMPLE POLYGON?
- Can our approach be extended to polygons with holes? A major obstacle is that in this case a triangulation using shortest paths not always exists. We have been able to generalize our approach, by working with a suitable *collection* of paths between every pair of sites, but this significantly complicates matters, thus defeating the purpose.
- Can ideas from our approach be used to get a simplified solution to the more general SUBSET TSP problem for planar graphs?
- What about TSP WITH OBSTACLES in higher dimensions? Here neither our approach nor the approach by Marx, Pilipczuk, and Pilipczuk can be used.

References

1 Jeff Abrahamson and Ali Shokoufandeh. Euclidean TSP on two polygons. *Theor. Comput. Sci.*, 411(7-9):1104–1114, 2010. `doi:10.1016/j.tcs.2009.12.003`.

2 David L. Applegate, Robert E. Bixby, and Vasek Chvátal. *Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

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

4 Sanjeev Arora, Michelangelo Grigni, David R. Karger, Philip N. Klein, and Andrzej Woloszyn. A polynomial-time approximation scheme for weighted planar graph TSP. In Howard J. Karloff, editor, *Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1998)*, pages 33–41. ACM/SIAM, 1998. URL: `http://dl.acm.org/citation.cfm?id=314613.314632`.

5 Yair Bartal, Lee-Ad Gottlieb, and Robert Krauthgamer. The traveling salesman problem: Low-dimensionality implies a polynomial time approximation scheme. *SIAM J. Comput.*, 45(4):1563–1581, 2016. `doi:10.1137/130913328`.

6 Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *J. ACM*, 9(1):61–63, 1962. doi:10.1145/321105.321111.

7 T.-H. Hubert Chan and Anupam Gupta. Approximating TSP on metrics with bounded global growth. *SIAM J. Comput.*, 41(3):587–617, 2012. doi:10.1137/090749396.

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

9 William J. Cook. *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton University Press, 2011.

10 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

11 Mark de Berg, Hans L. Bodlaender, Sándor Kisfaludi-Bak, and Sudeshna Kolay. An eth-tight exact algorithm for euclidean TSP. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 450–461, 2018. doi:10.1109/FOCS.2018.00050.

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

13 Marcus Greiff and Anders Robertsson. Optimisation-based motion planning with obstacles and priorities. *IFAC-PapersOnLine*, 50(1):11670–11676, 2017. 20th IFAC World Congress. doi:10.1016/j.ifacol.2017.08.1677.

14 Leonidas J. Guibas and John Hershberger. Optimal shortest path queries in a simple polygon. *Journal of Computer and System Sciences*, 39(2):126–152, 1989. doi:10.1016/0022-0000(89)90041-X.

15 Gregory Gutin and Abraham P. Punnen. *The Traveling Salesman Problem and Its Variations*. Springer, 2002.

16 Michael Held and Richard 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.

17 R. Z. Hwang, R. C. Chang, and Richard C. T. Lee. The searching over separators strategy to solve some NP-hard problems in subexponential time. *Algorithmica*, 9(4):398–423, 1993. doi:10.1007/BF01228511.

18 Viggo Kann. *On the approximability of NP-complete optimization problems*. PhD thesis, Royal Institute of Technology Stockholm, 1992.

19 Sándor Kisfaludi-Bak. A quasi-polynomial algorithm for well-spaced hyperbolic TSP. In *36th International Symposium on Computational Geometry (SoCG 2020)*, volume 164 of *LIPIcs*, pages 55:1–55:15, 2020. doi:10.4230/LIPIcs.SoCG.2020.55.

20 Sándor Kisfaludi-Bak, Jesper Nederlof, and Karol Wegrzycki. A gap-eth-tight approximation scheme for euclidean TSP. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 351–362. IEEE, 2021. doi:10.1109/FOCS52979.2021.00043.

21 Philip N. Klein. A linear-time approximation scheme for TSP in undirected planar graphs with edge-weights. *SIAM J. Comput.*, 37(6):1926–1952, 2008. doi:10.1137/060649562.

22 Philip N. Klein and Dániel Marx. A subexponential parameterized algorithm for subset TSP on planar graphs. In *Proc. 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2014)*, pages 1812–1830, 2014. doi:10.1137/1.9781611973402.131.

23 Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. On subexponential parameterized algorithms for steiner tree and directed subset TSP on planar graphs. *CoRR*, abs/1707.02190, 2017. arXiv:1707.02190.

24 Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. On subexponential parameterized algorithms for steiner tree and directed subset TSP on planar graphs. In *59th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2018)*, pages 474–484, 2018. doi:10.1109/FOCS.2018.00052.

25 Gary L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *Journal of Computer and System Sciences*, 32(3):265–279, 1986. doi:10.1016/0022-0000(86)90030-9.

26 Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.*, 28(4):1298–1309, 1999. doi:10.1137/S0097539796309764.

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

28 Satish Rao and Warren D. Smith. Approximating geometrical graphs via "spanners" and "banyans". In *STOC*, pages 540–550. ACM, 1998.

29 Gerhard Reinelt. *The Traveling Salesman, Computational Solutions for TSP Applications*, volume 840 of *Lecture Notes in Computer Science*. Springer, 1994. doi:10.1007/3-540-48661-5.

30 John Saalweachter and Zygmunt Pizlo. *Non-Euclidean Traveling Salesman Problem*, pages 339–358. Springer New York, New York, NY, 2008. doi:10.1007/978-0-387-77131-1_14.

31 Warren D. Smith and Nicholas C. Wormald. Geometric separator theorems & applications. In *FOCS*, pages 232–243. IEEE Computer Society, 1998. doi:10.1109/SFCS.1998.743449.

32 Godfried Toussaint. Oan optimal algorithm for computing the relative convex hull of a set of points in a polygon. In *Proceedings of EURASIP, Signal Processing III: Theories and Applications (Part 2)*, pages 853–856, 1986.

33 Luca Trevisan. When hamming meets euclid: The approximability of geometric TSP and steiner tree. *SIAM J. Comput.*, 30(2):475–485, 2000. doi:10.1137/S0097539799352735.