Approximate Algorithms for Touring a Sequence of Polygons

Fajie Li[1] and Reinhard Klette[2]

[1] Institute for Mathematics and Computing Science, University of Groningen
P.O. Box 800, 9700 AV Groningen, The Netherlands
[2] Computer Science Department, The University of Auckland
Private Bag 92019, Auckland 1142, New Zealand

Abstract. Given two points p and q and a finite number of simple polygons in a plane. The basic version of a touring-a-sequence-of-polygons problem (in brief: a *touring polygons problem*, TPP) is how to find a shortest path such that it starts at p, then it visits these polygons in the given order, and finally it ends at q. This paper describes approximate algorithms for different versions of touring polygons problems. Among its important results it provides, for example, an answer to the previously open problem "What is the complexity of the touring polygons problem for pairwise disjoint nonconvex simple polygons?" by providing a $\kappa(\varepsilon)$-linear approximate algorithm for solving this problem, with

$$\kappa(\varepsilon) = (L_0 - L_1)/\varepsilon$$

where L_0 is the length of the initial path and L is the true (i.e., optimum) path length. As a further example, this paper finds an approximate solution to the unconstrained touring polygons problem which is known to be NP-hard.

Key words: rubberband algorithm, Euclidean shortest path, ESP, touring polygons problem, TPP

1 Introduction

We recall notation from [2] for introducing a touring polygons problem (TPP). Let π be a plane, which is identified with $\mathbb{R}^2$. Consider polygons $P_i \subset \pi$, where $i = 1, 2, \ldots, k$, and two points p, $q \in \pi$. Let $p_0 = p$ and $p_{k+1} = q$. Let $p_i \in \mathbb{R}^2$, where $i = 1, 2, \ldots, k$. Let $\rho(p, p_1, p_2, , \ldots, p_k, q)$ denote the polygonal path $pp_1p_2 \ldots p_kq \subset \mathbb{R}^2$. Let $\rho(p, q) = \rho(p, p_1, p_2, , \ldots, p_k, q)$ if this does not cause any confusion. If $p_i \in P_i$ such that p_i is the first (i.e., along the path) point in $\partial P_i \cap \rho(p, p_i)$ (∂P is the frontier of P), then we say that path $\rho(p, q)$ *visits* P_i at p_i, where $i = 1, 2, \ldots, k$. The *unconstrained* TPP is defined as follows:

How to find a shortest path $\rho(p, p_1, p_2, , \ldots, p_k, q)$ such that it visits each of the polygons P_i in the given order $i = 1, 2, \ldots, k$?

Let $F_i \subset \mathbb{R}^2$ be a simple polygon such that $(P_i^\bullet \cup P_{i+1}^\bullet) \subset F_i^\bullet$ ($P^\bullet$ is the union of ∂P and the topological interior of P); then we say that F_i is a *fence* [with respect to P_i and P_{i+1} (mod $k+1$)], where $i = 0, 1, 2, \ldots, k+1$. Now assume that we have a fence F_i for any pair of polygons P_i and P_{i+1}, for $i = 0, 1, \ldots, k+1$. The *constrained* TPP is defined as follows:

How to find a shortest path $\rho(p, p_1, p_2, \ldots, p_k, q)$ such that it visits each of the polygons P_i in the given order, also satisfying $p_i p_{i+1}(mod\ k+1) \subset F_i^\bullet$, for $i = 1, 2, \ldots, k$?

Assume that for any i, $j \in \{1, 2, \ldots, k\}$, $\partial P_i \cap \partial P_j = \emptyset$, and each P_i is convex; this special case is dealt with in [2]. The given algorithm runs in $\mathcal{O}(kn\ log(n/k))$ time, where n is the total number of all vertices of all polygons $P_i \subset \pi$, for $i = 1, 2, \ldots, k$, and π. According to [2], "one of the most intriguing open problems" identified by their results "is to determine the complexity of the TPP for (pairwise) disjoint nonconvex simple polygons". Algorithm 4 in Section 3.1 answers this problem by providing an approximate algorithm running in time $\kappa(\varepsilon) \cdot \mathcal{O}(n)$, where n is the total number of vertices of all polygons. Most importantly, this paper provides an approximate solution (Theorem 3) to the unconstrained touring polygons problem (TPP) which is known to be NP-hard (see Theorem 1, cited from [2]).

2 Basics

We briefly recall some results used in the rest of this paper, starting with three algorithms and a characterization of a time complexity.

Algorithm 1 *2D ESP* (see [7], pages 639–641)
Input: *A simple polygon Π and two points p, $q \in \Pi^\bullet$.*
Output: *A set of vertices of a shortest path from p to q inside of $\Pi^\bullet$.*

Algorithm 2 *Convex Hull Algorithm* (see, e.g., [6] or Figure 13.7, [3])
Input: *A set of vertices of a simple polygon P.*
Output: *The set of vertices of the convex hull of P.*

Algorithm 3 *Tangent Calculation* (see [8])
Input: *A convex polygon P and a point $p \notin P^\bullet$.*
Output: *Two points $t_i \notin P^\bullet$ such that pt_i are tangents to P, where i = 1, 2.*

Theorem 1. (see [2], Theorem 6) *The touring polygons problem (TPP) is NP-hard, for any Minkowski metric L_p ($p \geq 1$) in the case of nonconvex polygons P_i, even in the unconstrained ($F_i = \mathbb{R}^2$) case with obstacles bounded by edges having angles 0, 45, or 90 degrees with respect to the x-axis.*

In Section 4, we will apply Algorithm 5 to show that the same problem stated in Theorem 1 can be approximately solved in $\kappa(\varepsilon)$ linear time.

In the following, let $V(P)$ be the set of vertices of polygon P and $E(P)$ the set of edges of P. $d_e(v, e)$ is the Euclidean distance between a point v and a segment e. Let $\kappa(\varepsilon) = (L_0 - L)/\varepsilon$ where L_0 is the length of the initial path and L is the true (i.e., optimum) path length.

3 The Algorithms

3.1 An Algorithm for the Unconstrained TPP

Our first algorithm (Algorithm 4) answers the open problem in Section 1 by an approximate algorithm. Let π be a plane, containing the polygon $V(P_i) \subset \pi$, where $i = 1, 2, \ldots, k$. Suppose that, for any i, $j \in \{1, 2, \ldots, k\}$, $\partial P_i \cap \partial P_j = \emptyset$. Let p, $q \in \pi$, $p = p_0$, and $q = p_{k+1}$.

Algorithm 4 *Unconstrained TPP (for pairwise disjoint polygons)*
Input: *k disjoint polygons P_1, P_2, …, P_k (i.e., $\partial P_i \cap \partial P_{i+1} = \emptyset$, i = 1, 2, …, $k - 1$); two points p, $q \notin \partial P_i$, where i = 1, 2, …, k.*
Output: *The set of vertices of the shortest path which starts at p, then visits P_i in order, and finally ends at q.*

1: Let $\varepsilon = 10^{-10}$ (i.e., an example of a chosen accuracy).
2: **for** each $i \in \{1, 2, \ldots, k\}$ **do**
3:     Let p_i be a point on ∂P_i.
4: **end for**
5: Compute the length L_0 of the path $\rho = \langle p, p_1, p_2, \ldots, p_k, q\rangle$.
6: Let $q_1 = p$ and $i = 1$.
7: **while** $i < k - 1$ **do**
8:     Let $q_3 = p_{i+1}$.
9:     Compute a point $q_2 \in \partial P_i$ (see Lemmas 1 and 2, and Theorem 1, all in [5]) such that
$d_e(q_1, q_2) + d_e(q_3, q_2) = \min\{d_e(q_1, q') + d_e(q_3, q') : q' \in \partial P_i\}$.
10:     Update ρ by replacing p_i by q_2.
11:     Let $q_1 = p_i$ and $i = i + 1$.
12: **end while**
13: Let $q_3 = q$.
14: Compute $q_2 \in \partial P_k$ such that
$d_e(q_1, q_2) + d_e(q_3, q_2) = \min\{d_e(q_1, q) + d_e(q_3, q) : q \in \partial P_k\}$.
15: Update ρ by replacing p_k by q_2.
16: Compute the length L_1 of the updated path $\rho = \langle p, p_1, p_2, \ldots, p_k, q\rangle$.
17: Let $\delta = L_0 - L_1$.
18: **if** $\delta > \varepsilon$ **then**
19:     Let $L_0 = L_1$ and go to Step 6.
20: **else**
21:     Output $\{p, p_1, p_2, \ldots, p_k, q\}$ and Stop.
22: **end if**

The correctness of this algorithm follows by

Theorem 2. *The solution obtained by Algorithm 4 is an approximate global solution to the touring polygons problem (without joint polygons).*

Proof. This proof is a modification of the one given for Theorem 2 in [5]. Let $X = \Pi_{i=1}^{k} \partial P_i$; ∂P_i is as defined in Algorithm 4. Let Y be the set of all solutions obtained by Algorithm 4. By Lemmas 1, 2, and Theorem 1 of [5], Algorithm 4 defines a continous function, denoted by f, mapping from X to Y depending on the chosen accuracy $\varepsilon > 0$.

Note that, if each ∂P_i is degenerated into a single edge, then there exists a unique solution to the ESP problem (see [1, 9, 11]). Now, let $v = (v_1, v_2, \ldots, v_k) \in Y$. Because of $v_i \in \partial P_i$, where $i = 1, 2, \ldots, k$, it follows that Y is a finite set.

Now we prove that Y is a singleton. Otherwise, take $v_0 \in Y$, then we have $f^{-1}(v_0) \subset X$. For each $v \in f^{-1}(v_0)$, as f is a continous function, there exists a sufficiently small open neighborhood (with respect to the Euclidean topology on X) of v, denoted by $N(v, \delta_v)$, such that for each $v' \in N(v, \delta_v)$, $f(v') = v_0$. Thus, $N(v, \delta_v) \subseteq f^{-1}(v_0)$ and $\cup_{v\in f^{-1}(v_0)} N(v, \delta_v) \subseteq f^{-1}(v_0)$. On the other hand, as $f^{-1}(v_0) = \{v : v \in f^{-1}(v_0)\}$ and $v \in N(v, \delta_v)$, thus we have that $f^{-1}(v_0) \subseteq \cup_{v\in f^{-1}(v_0)} N(v, \delta_v)$. Therefore, $f^{-1}(v_0) = \cup_{v\in f^{-1}(v_0)} N(v, \delta_v)$. As $N(v, \delta_v)$ is an open set, $f^{-1}(v_0)$ is an open set as well. Let $f^{-1}(v_0) = \cup_{i=1}^{k} S_i$, where S_i is an open subset of ∂P_i, $i = 1, 2, \ldots, k$. Recall that $f^{-1}(v_0) \subset X$, so there exists a S_i such that $\emptyset \subset S_i \subset \partial P_i$. Without loss of generality, suppose that $\emptyset \subset S_1 \subset \partial P_1$. Now, S_1 is a nonempty open subset of ∂P_1. S_1 is a union of a countable number of open intervals (Proposition 5.1.4, [10]). Thus, there exists a point $w_1 \in \partial P_1 \setminus S_1$ such that, for each positive ε_1, there exists a point $w_1' \in N(w_1, \varepsilon_1) \cap S_1$ [again, $N(w_1, \varepsilon_1)$ is an open neighborhood with respect to the usual topology on ∂P_1]. Therefore, there exists a point $v_1 \in X \setminus f^{-1}(v_0)$ such that, for each positive ε_1, there exists a point $v_1' \in N(v_1, \varepsilon_1) \cap f^{-1}(v_0)$. This contradicts that f is a continous function on X. Thus, Y is a singleton. □

The main idea of the proof is as follows: every considered (i.e., defined by the ESP problem environment!) continuous function with a finite number of local minima should have a unique minimum.

Note that Algorithm 4 still works if the input polygons are not in the same plane (even if the edges of a single polygon are not in the same plane). Also, the input polygons do not have to be simple.

The following procedure handles the case when polygons are not (pairwise) disjoint, by "slightly" modifying one of the polygons (see Figure 1).

Procedure 1 *Case of overlapping polygons*
Input: *A point p and two polygons P_1 and P_2 such that $p \in \partial P_1 \cap \partial P_2$.*
Output: *A point $q \notin \partial P_2$ (and a "slightly" updated P_1 such that it does not intersect with P_2 at point p).*

1: Let $\varepsilon = 10^{-10}$ (i.e., an assumed accuracy).
2: Find a point $e_j \in E(P_j)$, where $j = 1, 2$, such that $p \in e_1 \cap e_2$.
3: Let $e_1 = q_1q_2$. Let q_3 and q_4 be two points in two segments q_1p and q_2p, respectively (see Figure 1) such that $d_e(q_j, p) \le \varepsilon'$ and $q_j \notin \partial P_2$, where $j = 3, 4$.
4: Find two points q_3' and q_4' such that $q_3'q_3$ and $q_4'q_4 \perp$ the plane defined by e_1 and e_2, and $d_e(q_3', q_3) = d_e(q_4', q_4) = 2 \times \varepsilon'$.
5: Slightly update P_1 by replacing the edge $e_1 = q_1q_2$ by polyline $q_1q_3q_3'q_4'q_4q_2$.
6: Output q_3'.

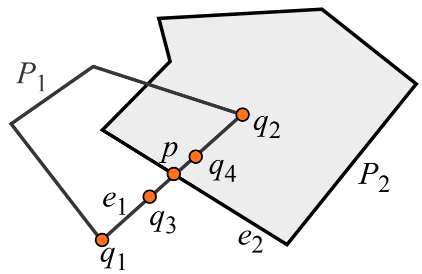

Fig. 1. Illustration for Procedure 1.

If there exist i, $j \in \{1, 2, \ldots, k\}$ such that $i \neq j$ and $\partial P_i \cap \partial P_{i+1} \neq \emptyset$, then we modify Algorithm 4 as follows: apply Procedure 1 after Steps 7 and 12.

Algorithm 5 *Unconstrained TPP (polygons do not have to be pairwise disjoint)*
Input: *k polygons P_1, P_2, ..., P_k; two points p, $q \notin \partial P_i$, where i = 1, 2, ..., k.*
Output: *The set of vertices of the shortest path which starts at p, then visits P_i in order, and finally ends at q.*

1: Let $\varepsilon = 10^{-10}$ (the accuracy).
2: **for** each $i \in \{1, 2, \ldots, k\}$ **do**
3: Let p_i be a point on ∂P_i.
4: **end for**
5: Compute the length L_0 of the path $\rho = \langle p, p_1, p_2, \ldots, p_k, q\rangle$.
6: Let $q_1 = p$ and $i = 1$.
7: **while** $i < k - 1$ **do**
8: **if** $(p_i = p_{i-1} \wedge p_i \neq p_{i+1}) \vee (p_i \neq p_{i-1} \wedge p_i = p_{i+1}) \vee (p_i = p_{i-1} \wedge p_i = p_{i+1})$ **then**
9: Apply Procedure 1 to compute a point p_i such that $p_i \neq p_{i-1}$ and $p_i \neq p_{i+1}$.
10: **end if**
11: Let $q_3 = p_{i+1}$.
12: Compute a point $q_2 \in \partial P_i$ such that
$d_e(q_1, q_2) + d_e(q_3, q_2) = \min\{d_e(q_1, q') + d_e(q_3, q') : q' \in \partial P_i\}$.
13: Update ρ by replacing p_i by q_2.
14: Let $q_1 = p_i$ and $i = i + 1$.
15: **end while**
16: **if** $(p_k = p_{k-1} \wedge p_k \neq p_{k+1}) \vee (p_k \neq p_{k-1} \wedge p_k = p_{k+1}) \vee (p_k = p_{k-1} \wedge p_k = p_{k+1})$ **then**
17: Apply Procedure 1 to compute a point p_k such that $p_k \neq p_{k-1}$ and $p_k \neq p_{k+1}$.
18: **end if**
19: Let $q_3 = q$.
20: Compute $q_2 \in \partial P_k$ such that
$d_e(q_1, q_2) + d_e(q_3, q_2) = \min\{d_e(q_1, q') + d_e(q_3, q') : q' \in \partial P_k\}$.

21: Update P by replacing p_k by q_2.
22: Compute the length L_1 of the updated path $\rho = \langle p, p_1, p_2, \ldots, p_k, q\rangle$.
23: Let $\delta = L_0 - L_1$.
24: **if** $\delta > \varepsilon$ **then**
25: Let $L_0 = L_1$ and go to Step 6.
26: **else**
27: Output $\{p, p_1, p_2, \ldots, p_k, q\}$ and Stop.
28: **end if**

Basically following the same way as provided with the proof of Theorem 2, we proved that Algorithm 5 computes an approximate global solution to the unconstrained TPP (but will not provide here due to given similarities).

Section 4 applies this algorithm to show that the TPP, for not necessarily pairwise disjoint, and not necessarily convex, simple polygons can be approximately computed in $\kappa(\varepsilon)$ linear time. By Theorem 1, finding the exact solution of this problem is NP-hard.

3.2 Algorithms for Solving the Constrained TPP

The following Procedure compute a "maximal" polyline in the frontier of a small polygon P such that it "can be seen" by a vertex of a big polygon Π. It will be used in Steps 9 and 10 in the next procedure.

Procedure 2 Input: *Two simple polygons* Π *and* P *such that* $P^\bullet \subset \Pi^\bullet$*; two points* $q \in \partial\Pi$ *and* $p \in \partial P$ *such that there exist two points* p_1*,* $p_2 \in \partial P$ *such that* $pp_i \subset \partial P$*, and* $d_e(p, p_i)$ *is a sufficiently small positive number, and* $\triangle qpp_i \subset \Pi^\bullet$*, where* i *= 1, 2.*
Output: *Two points* p'_1*,* $p'_2 \in \partial P$ *such that* $pp'_i \subset \partial P$*, and* $d_e(p, p'_i)$ *is a positive number as large as possible such that* $\triangle qpp'_i \subset \Pi^\bullet$*, where* i *= 1, 2.*

1: **if** $p \notin V(P)$ **then**
2: There exists an edge $e = v_1v_2 \in E(P)$ such that $p \in e$ and $d_e(p, v_i) > 0$, $i = 1, 2$; select such an edge for the following.
3: **if** q, v_1 and v_2 are colinear (see top left, Figure 2) **then**
4: Let $p'_i = v_i$.
5: **else**
6: **for** $i \in \{1, 2\}$ **do**
7: **if** $qv_i \cap \partial\Pi = \emptyset$ (see top middle, Figure 2) **then**
8: Let $p'_i = v_i$.
9: **else**
10: Apply Algorithm 2 and Algorithm 3 to find a point $p'_i \in e$ such that qp'_i is a tangent to Π (see top right, Figure 2).
11: **end if**
12: **end for**
13: **end if**
14: **else**

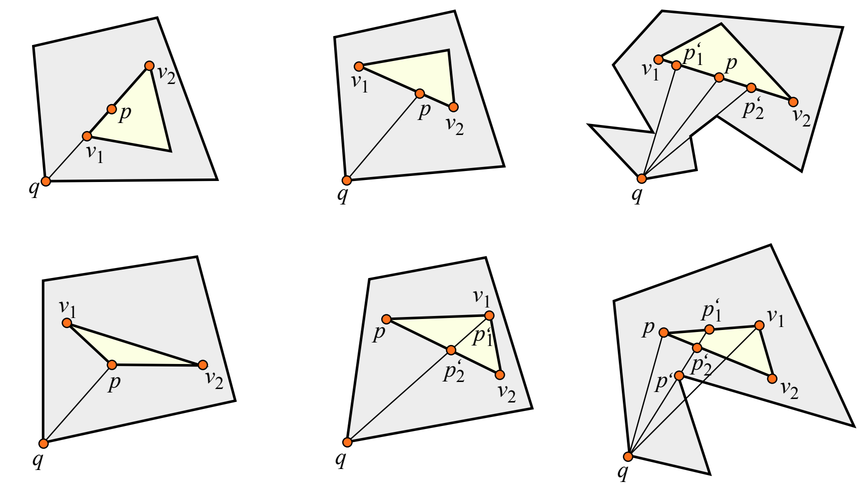

Fig. 2. Illustration for Procedure 2: Π is shown as the larger simple polygon, and P is the (smaller) triangle inside of Π.

15: There exist two points $v_i \in E(P)$ such that $pv_i \in E(P)$; select such points.
16: **if** v_1 and v_2 are on different sides of qp (bottom left, Figure 2) **then**
17: Proceed analogous to Steps 6–12.
18: **else**
19: Find $v \in \{v_1, v_2\}$ such that $d_e(v, qp) = \min\{d_e(v', qp) : v' = v_1 \vee v' = v_2\}$.
20: **if** $qv \cap \partial\Pi = \emptyset$ (see bottom middle, Figure 2) **then**
21: Let $p'_i = qv \cap pv_i$, where $i = 1, 2$.
22: **else**
23: Apply Algorithm 2 and Algorithm 3 to find a point p' such that qp' is a tangent to Π (see bottom right, Figure 2).
24: Let $p'_i = qp' \cap pv_i$, where $i = 1, 2$.
25: **end if**
26: **end if**
27: **end if**
28: Output p'_i, $i = 1, 2$, and Stop.

The following Procedure will be frequently used in Step 7 of the general TPP algorithm (Algorithm 6). It computes a local shortest path approximately.

Procedure 3 Input: *Three polygons P_1, P_2 and P_3 in that order, the fence of P_1 and P_2, denoted by F_{12}, the fence of P_2 and P_3, denoted by F_{23}, and three points $p_i \in \partial P_i$, where i = 1, 2, 3.*
Output: *The set of all vertices of the (approximate) shortest path which starts at p_1, then visits P_2, and finally ends at p_3 (see Figure 3).*

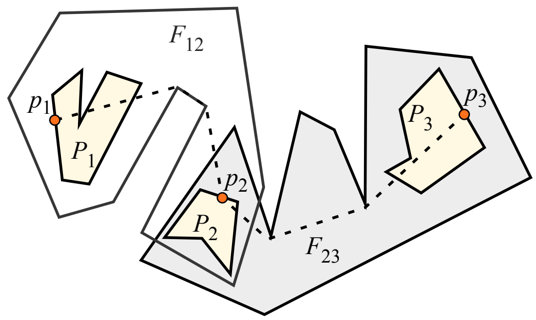

Fig. 3. Illustration for Procedure 3.

1: Let $\varepsilon = 10^{-10}$ (the accuracy)
2: **if** $(p_2 = p_1 \wedge p_2 \neq p_3) \vee (p_2 \neq p_1 \wedge p_2 = p_3) \vee (p_2 = p_1 \wedge p_2 = p_3)$ **then**
3: Apply Procedure 1 to compute a point which updates p_2 such that $p_2 \neq p_1$ and $p_2 \neq p_3$.
4: **end if**
5: Let p_1, p_2 and F_{12} be the input for Algorithm 1 to compute a shortest path from p_1 to p_2 inside of F_{12}, denoted by ρ_{12}.
6: Let p_2, p_3 and F_{23} be the input for Algorithm 1 to compute a shortest path from p_2 to p_3 inside of F_{23}, denoted by ρ_{23}.
7: Let $V = V(\rho_{12}) \cup V(\rho_{23})$.
8: Find q_1 and $q_3 \in V$ such that $\{q_1, p_2, q_3\}$ is a subsequence of V.
9: Let F_{12}, P_2, q_1, and p_2 be the input for Procedure 2 to compute a polyline, denoted by $v_1 p_2 v_2$.
10: Let F_{23}, P_2, q_3, and p_2 be the input for Procedure 2 to compute a polyline, denoted by $u_1 p_2 u_2$.
11: Let $s = v_1 p_2 v_2 \cap u_1 p_2 u_2$.
12: Find a point $p_2' \in \partial s$ such that
$$d_e(q_1, p_2') + d_e(p_2', q_3) = \min\{d_e(q_1, p') + d_e(p', q_3) : p' \in \partial s\}.$$
13: Update set V by letting $p_2 = p_2'$.
14: Output V.

Note that in Step 3, the updated point p_2 depends on the chosen value of ε'. The following is the main algorithm in this paper:

Algorithm 6 *Constrained TPP*
Input: *k polygons P_i; k polygons F_i which are the fence of P_i and P_{i+1} (mod k), where i = 0, 1, ..., $k-1$.*
Output: *The set of vertices of the shortest path $\rho = (p_0, p_1, \ldots, p_{k-1}, p_0)$ such that $p_i \in \partial P_i$, where i = 0, 1, ..., $k-1$; and L_1, its calculated length.*

1: **for** each $i \in \{0, 1, \ldots, k-1\}$ **do**
2: Let p_i be a point on ∂P_i.

3: **end for**
4: Let $V = \{p_0, p_1, \ldots, p_{k-1}\}$.
5: Calculate the perimeter L_0 of that polygon which has V as its set of vertices (in the given order).
6: **for** each $i \in \{0, 1, \ldots, k-1\}$ **do**
7:    Use P_{i-1}, P_i, P_{i+1} (mod k) and F_{i-1}, F_i (mod k) as input for Procedure 3 for updating p_i and for calculating set V_i.
8:    Let $V' = V$ and update V by replacing $\{p_{i-1}, \ldots, p_i, \ldots, p_{i+1}\}$ by V_i.
9: **end for**
10: Let $V = \{q_0, q_1, \ldots, q_m\}$.
11: Calculate the perimeter L_1 of the polygon which has V as its set of vertices.
12: **if** $L_0 - L_1 > \varepsilon$ **then**
13:    Let $L_0 = L_1$, $V = V'$, and go to Step 4.
14: **else**
15:    Output the updated set V and (its) calculated length L_1.
16: **end if**

Analogous to the proof of Theorem 2, we proved that Algorithm 6 computes an approximate global solution to the constrained TPP (but do not provide the proof here due to similarities).

4 Computational Complexity

This section analyzes (step by step) the time complexity of procedures and algorithms presented above in this paper.

4.1 Unconstrained TPP

Lemma 1. *Algorithm 4 has a time complexity in* $\kappa(\varepsilon) \cdot \mathcal{O}(\sum_{j=1}^{k} |V(P_j)|)$.

Proof. Steps 1, 6, 8, 10, 11, 13, 15, 17, 19 only require constant time. Steps 2–5, 16, 21 can be computed in $\mathcal{O}(k)$ time. By the proofs of Lemmas 1, 2 and Theorem 1, [5], Steps 9 and 14 can be computed in $\mathcal{O}(|V(P_j)|)$ time, where $V(P_j)$ is as in Algorithm 4, for $j = i, k$. Thus, each iteration of Algorithm 4 can be computed in time $\mathcal{O}(\sum_{j=1}^{k} |V(P_j)|)$. Therefore, Algorithm 4 can be computed in $\kappa(\varepsilon) \cdot \mathcal{O}(\sum_{j=1}^{k} |V(P_j)|)$ time. This proves the lemma. □

Lemma 2. *Procedure 1 can be computed in* $\mathcal{O}(|E(P_1)| + |E(P_2)|)$ *time.*

Proof. Steps 1, 4, 5, and 6 only need constant time. Step 2 can be computed in time $\mathcal{O}(|E(P_1)|+|E(P_2)|)$, and Step 3 in time $\mathcal{O}(|E(P_2)|)$. Therefore, Procedure 1 can be computed in $\mathcal{O}(|E(P_1)| + |E(P_2)|)$ time. □

Lemma 3. *Algorithm 5 can be computed in time* $\kappa(\varepsilon) \cdot \mathcal{O}(\sum_{j=1}^{k} |E(P_j)|)$.

Proof. The difference between Algorithm 5 and Algorithm 4 is defined by Steps 8–10 and 16–18. By Lemma 2, Steps 8–10 and 16–18 can be computed in $\mathcal{O}(|E(P_{j-1})|+2|E(P_j)|+|E(P_{j+1})|)$ time, where $j = i, k$. Thus, each iteration of Algorithm 5 can be computed in time $\mathcal{O}(\sum_{j=1}^{k}|E(P_j)|)$ Therefore, Algorithm 5 can be computed in $\kappa(\varepsilon)\cdot\mathcal{O}(\sum_{j=1}^{k}|E(P_j)|)$ time. This proves the lemma. □

By Lemmas 1 and 3, we have the following

Theorem 3. *The unconstrained TPP can be solved approximately in* $\kappa(\varepsilon)\cdot\mathcal{O}(n)$ *time, where* n *is the total number of vertices of all polygons involved.*

4.2 Constrained TPP

Lemma 4. *Procedure 2 can be computed in* $\mathcal{O}(|V(\Pi)|+|V(P)|)$ *time.*

Proof. Step 1 can be computed in $\mathcal{O}(|V(P)|)$ time. Step 2 can be computed in $\mathcal{O}(|E(P)|) = \mathcal{O}(|V(P)|)$ time. Steps 3–8 only need constant time. Steps 10 and 23 can be computed in $\mathcal{O}(|V(\Pi)|)$ time (see [6], [8]).

Step 15 can be computed in $\mathcal{O}(|E(P)|)$ time. Steps 16 and 17 can be computed in $\mathcal{O}(|V(\Pi)|)$ time. Steps 19–21, 24 and 28 only need constant time.

Altogether, the time complexity of Procedure 2 is $\mathcal{O}(|V(\Pi)|+|V(P)|)$. □

Lemma 5. *Procedure 3 can be computed in time*
$\mathcal{O}(|E(P_1)|+2|E(P_2)|+|E(P_3)|+|E(F_{12})|+|E(F_{23})|)$.

Proof. Step 1 requires only constant time. By Lemma 2, Steps 2–4 can be computed in time $\mathcal{O}(|E(P_1)|+2|E(P_2)|+|E(P_3)|)$. Step 5 can be computed in $\mathcal{O}(|V(F_{12})|)$ (see [7]). Step 6 can be computed in $\mathcal{O}(|V(F_{23})|)$. Step 7 in $\mathcal{O}(|V(F_{12})|+|V(F_{23})|))$ time. Step 8 in $\mathcal{O}(|V|))$ time. By Lemma 4, Step 9 can be computed in $\mathcal{O}(|V(F_{12})|+|V(P_2)|)$; Step 10 can be computed in $\mathcal{O}(|V(F_{23})|+|V(P_2)|)$. Steps 11–13 require only constant time. Step 14 is in $\mathcal{O}(|V|))$ time.

Therefore, Procedure 3 can be computed in $\mathcal{O}(|E(P_1)|+2|E(P_2)|+|E(P_3)|+|E(F_{12})|+|E(F_{23})|)$ time. This proves the lemma. □

Lemma 6. *Algorithm 6 can be computed in time* $\kappa(\varepsilon)\cdot\mathcal{O}(n)$*, where* n *is the total number of all vertices of the polygons involved.*

Proof. Steps 1–5 can be computed in $\mathcal{O}(k)$ time. By Lemma 5, each iteration in Steps 7 and 8 can be computed in time $\mathcal{O}(\sum_{i=0}^{k-1}(|E(P_{i-1})|+2|E(P_i)|+|E(P_{i+1})|+|E(F_{i-1})|+|E(F_i)|))$. Steps 10–16 can be computed in $\mathcal{O}(|V|)$. Note that $|V|\le\sum_{i=0}^{k-1}(|V(P_i)|+|V(F_i)|)$, $|V(P_i)| = |E(P_i)|$, and $|V(F_i)| = |E(F_i)|$, where $i = 0, 1, \ldots, k-1$. Therefore, Algorithm 6 can be computed in $\kappa(\varepsilon)\cdot\mathcal{O}(\sum_{i=0}^{k-1}(|E(P_{i-1})|+2|E(P_i)|+|E(P_{i+1})|+|E(F_{i-1})|+|E(F_i)|))$ time, where each index is taken mod k. This time complexity is equivalent to $\kappa(\varepsilon)\cdot\mathcal{O}(n)$ where n is the total number of vertices of all polygons. □

Lemma 6 allows to conclude the following

Theorem 4. *The constrained TPP can be solved approximately in $\kappa(\varepsilon) \cdot \mathcal{O}(n)$ time, where n is the total number of all vertices of involved polygons.*

According to Theorem 1, Theorem 4 is the best possible result in some sense.

5 Conclusions

The paper started with recalling an open problem published in 2003 in [2], that "one of the most intriguing open problems ... is to determine the complexity of the TPP for (pairwise) disjoint nonconvex simple polygons". The paper described a simple rubberband algorithm ([4]) which "approximately" answers this open problem.

Note that the solution in [2] is only valid if the following two requirements are satisfied: the polygons should be convex and pairwise disjoint; the given algorithm has time complexity $\kappa(\epsilon) \cdot \mathcal{O}(kn\, log(n/k))$, where n is the total number of vertices of involved polygons $P_i \subset \pi$, for $i = 1, 2, \ldots, k$.

The algorithm presented in this paper also applies to nonconvex polygons, even non-simple polygons, and polygons whose edges do not have to be in the same plane, and it is of κ-linear time complexity. An important result in this paper is Theorem 3 which provides an approximate solution to the unconstrained touring polygons problem (TPP) which is known to be NP-hard (see the cited Theorem 1).

References

1. J. Choi, J. Sellen, and C.-K. Yap. Precision-sensitive Euclidean shortest path in 3-space. In Proc. *Annu. ACM Sympos. Computational Geometry*, pages 350–359, 1995.
2. M. Dror, A. Efrat, A. Lubiw, and J. Mitchell. Touring a sequence of polygons. In Proc. *STOC*, pages 473–482, 2003.
3. R. Klette and A. Rosenfeld. *Digital Geometry.* Morgan Kaufmann, San Francisco, 2004.
4. F. Li and R. Klette. Rubberband algorithms for solving various 2D or 3D shortest path problems. In Proc. *Computing: Theory and Applications*, The Indian Statistical Institute, Kolkata, pages 9 - 18, IEEE, 2007.
5. F. Li and R. Klette. Approximate Shortest Path Calculations in Simple Polyhedra. MI-techTR 23, The University of Auckland, 2008 (`http://www.mi.auckland.ac.nz/index.php?option=com\_content\&view=\\article\&id=91\&Itemid=76`).
6. A. Melkman. On-line construction of the convex hull of a simple polygon. *Information Processing Letters*, **25**:11–12, 1987.
7. J. S. B. Mitchell. Geometric shortest paths and network optimization. In *Handbook of Computational Geometry* (J.-R. Sack and J. Urrutia, editors). pages 633–701, Elsevier, 2000.
8. D. Sunday. Algorithm 14: Tangents to and between polygons. See `http://softsurfer.com/Archive/algorithm_0201/` (last visit: November 2008).

9. M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. *SIAM J. Comput.*, **15**:193–215, 1986.
10. B. G. Wachsmuth. Interactive real analysis. See `http://web01.shu.edu/projects/reals/topo/index.html` (last visit: October, 2008)
11. C.-K. Yap. Towards exact geometric computation. *Computational Geometry: Theory Applications.*, **7**:3–23, 1997.