Approximate Shortest Path Algorithms for Sequences of Pairwise Disjoint Simple Polygons

Xiuxia Pan, Fajie Li* Reinhard Klette†

Abstract

Assume that two points p and q are given and a finite ordered set of simple polygons, all in the same plane; the basic version of a touring-a-sequence-of-polygons problem (TPP) is to find a shortest path such that it starts at p, then visits these polygons in the given order, and ends at q. This paper describes four approximation algorithms for unconstrained versions of problems defined by touring an ordered set of polygons. It contributes to an approximate and partial answer to the previously open problem "What is the complexity of the touring-polygons problem for pairwise disjoint, simple and not necessarily convex polygons?" by providing $\kappa(\varepsilon)\mathcal{O}(n)$ approximation algorithms for solving this problem, either for given start and end points p and q, or with allowing to have those variable, where n is the total number of vertices of the given k simple and pairwise disjoint polygons; $\kappa(\varepsilon)$ defines the numerical accuracy in dependency of a selected $\varepsilon > 0$.

1 TPP - Introduction and Review

The touring polygon problem (TPP), as defined in the abstract of this paper, does have various applications such as for parts cutting in the industry (e.g., moving the head of a cutting robot to subsequent start positions of planar shapes) or for route planning in general when the task consists in visiting selected polygonal regions. The TPP already has a history of publications addressing this subject; see, for example, [1, 2, 7, 11, 15].

1.1 Variants of the Touring Polygon Problem

We recall some notations from [2]; this paper introduced the TPP. Let π be a plane. Consider simple, pairwise disjoint polygons $P_i \subset \pi$, where $i = 1, 2, \ldots, k$, and two points p, $q \in \pi \setminus \bigcup_{i=1}^{k} P_i$. Note that a simple polygon P is a planar region whose frontier ∂P is normally represented by a polygonal path (i.e., a sequence of line segments whose endpoints define the vertices of P).

*College of Computer Science and Technology, Huaqiao University, Xiamen, Fujian, China, {panpanty, li.fajie}@yahoo.com

†Computer Science Department, The University of Auckland, Private Bag 92019, Auckland 1142, New Zealand, r.klette@auckland.ac.nz

Let $\rho(p, p_1, p_2, , \ldots, p_k, q)$ be a polygonal path in π that starts at $p_0 = p$, is then incident with points p_i in the given order, and ends at $p_{k+1} = q$, with $p_i \in \pi$, for $i = 1, 2, \ldots, k$. We denote such a path briefly by $\rho(p, q)$ if this does not cause any confusion (i.e., if intermediate points p_i are known by context or not important to be listed).

A path $\rho(p, q)$ *visits a polygon* R at point $r \in R$ if the path intersects R, and r is the first (i.e., along the path) point in R on this path. Obviously, $r \in \partial R$. The (unconstrained) *fixed TPP* is defined as follows:

> 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 at point p_i, for $i = 1, 2, \ldots, k$.

The path may be further constrained by some predefined properties, and we discuss examples below. – In case that start and end points of the path are not given then the problem becomes the (unconstrained) *floating TPP*:

> Let $P_0 = P_k$. Find a cyclic shortest path $\rho(p_0, p_1, p_2, , \ldots, p_{k-1}, p_k)$, with $p_0 = p_k$, such that it visits each of the polygons P_i in the given order at point p_i, for $i = 0, 1, 2, \ldots, k$.

Obviously, in this case it does not matter at which polygon to start, and we could reorder the polygons modulo k.

We just mention an example of further constraints (without discussing it further in this paper). Let $F \subset \pi$ be a simple, not necessarily convex polygon which contains the union of some polygons, all given in the plane π. Then F is called a *fence* with respect to those polygons.

In our case, we have polygons $P_1, \ldots, P_k$. Let $F_i \subset \pi$ be a fence for polygons P_i and P_{i+1}, for $i = 0, 1, 2, \ldots, k$, with degenerated polygons $P_0 = \{p = p_0\}$ and $P_{k+1} = \{q = p_{k+1}\}$. An example of a *constrained fixed TPP* is defined as follows:

> 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 at point p_i, for $i = 1, 2, \ldots, k$, also satisfying that F_i contains the subpath from p_i to p_{i+1}, for $i = 0, 1, 2, \ldots, k$.

The subpath from p_i to p_{i+1}, for $i = 0, 1, 2, \ldots, k$, needs to be a shortest path within a given polygon F_i. A shortest path, connecting two given points within a simple polygon, and fully contained in this polygon, can be constructed in time linear in the number of vertices of this polygon [14].

1.2 Variants of the Parts Cutting Problem

In a variety of industries, such as clothing, window manufacturing, or metal sheet processing, it is necessary to cut a set of parts (modeled by polygons) from large sheets of paper, cloth, glass, metal, and so forth. Motivated by such applications, [7] introduced the following three models of cutting scenarios:

- *Continuous cutting.* The path of the cutting tool visits each object (i.e., polygon) to be cut just once. The tool can engage the object at any point on its frontier, but must cut the entire object before it travels to the next object. Accordingly, the same frontier point must be used for entry and departure from the object.
- *Endpoint cutting.* The tool can enter and exit the object only at some predefined frontier points; however, it may cut the object in sections (i.e., it may visit an object repeatedly).
- *Intermittent cutting.* This is the most general version of the problem in which the object can be cut in sections and there is no restriction on the frontier points that can be used for entry or exit.

[7] focused on solving the continuous cutting problem where each object is a polygon, also called the *plate-cutting traveling salesman problem* (P-TSP). The P-TSP is a generalization of the well-known *traveling salesman problem* (TSP) [10]. The P-TSP is more general than the *generalized TSP* (GTSP) as discussed in [9, 15]. If each polygon degenerates into a single vertex then the P-TSP becomes the TSP which is known to be NP-hard [3]. It follows that the P-TSP is NP-hard as well.

The *ordered P-TSP* is a simplified TSP by providing a predefined order of visits; see Fig. 1. The polygons may still be nonconvex as well. [7] solved the ordered P-TSP by a heuristic approach based on a Lagrange relaxation method as discussed in [4, 5], without providing a time complexity analysis for this proposed approach.

As a follow-up of this work, [1] proved that a further simplified ordered P-TSP, where polygons are all assumed to be convex (see Figure 2), is solvable in polynomial time. If, additionally, the start point is also given (see Figure 3), then the authors of [2] claim that they can solve this fixed problem in time $\mathcal{O}(kn \log(n/k))$, where n is the total number of vertices of polygons

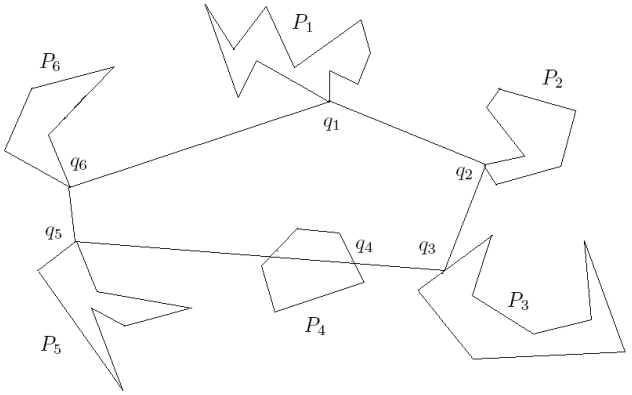

Figure 1: The ordered P-TSP in [7]. Not necessarily convex polygons are assumed to be given in a particular order. The cutting tool may travel across the given polygons.

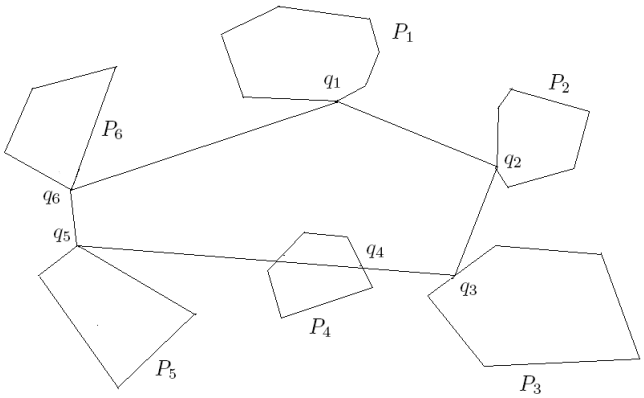

Figure 2: The ordered P-TSP in [1], also assuming that all polygons are convex.

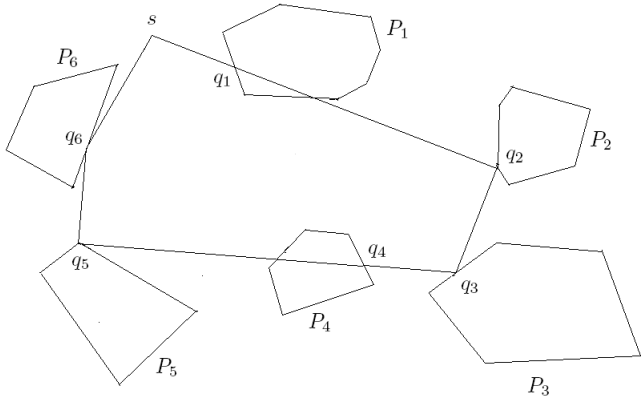

Figure 3: The P-TSP as considered in [2], now also with a given start point s.

$P_i \subset \pi$, for $i = 1, 2, \ldots, k$. Obviously, this fixed ordered P-TSP coincides with the fixed TPP as defined earlier; in this paper we prefer to use the naming based on touring polygons rather than on a traveling salesman.

2 Contributions of this Paper

According to [2], "one of the most intriguing open problems" identified by their results "is to determine the complexity of the fixed TPP for pairwise disjoint nonconvex simple polygons". In this paper, we focus on the unconstrained *fixed TPP* (i.e., given start and end point of the path) and *floating TPP* (i.e., no given start or end point) under the condition that the convex hulls of the input polygons P_i are pairwise disjoint, but the polygons P_i itself may be nonconvex.

Algorithm 2 in Section 3 partially answers the stated open problem for the fixed TPP by providing an approximation algorithm running in time $\kappa(\varepsilon) \cdot \mathcal{O}(n)$, where n is the total number of vertices of all polygons. The solution technique proposed in [2] can only handle the fixed TPP, the fixed safari problem, and the fixed watchman route problem, all for convex polygons only. Our solution technique is suitable for solving both the fixed and the floating TPP with the same time complexity, also allowing nonconvex polygons P_i with pairwise disjoint convex hulls. (Our method might also be useful for solving the floating watchman route, the floating safari problem, and the floating zookeeper problem; but this is not a subject in this paper.)

Regarding approximation (or approximate) algorithms, we refer to the following definition as given, for example, in [6]: An algorithm is an *δ-approximation algorithm* for a minimization problem P iff, for each input instance I of P, the algorithm delivers a solution that is at most δ times the optimum solution.

Our approximate algorithms are based on the idea of a *rubberband algorithm* (RBA); for example, see [11], for variants of RBAs. Basically, such an algorithm starts with an *initial path* through the provided sequence of *step sets* (here: polygons P_i), and runs then in iterations through those (possibly "adjusted") sequence again while reducing (compared to the previous run) the length of the current path in each run. The important issue is to guarantee that the resulting Cauchy sequence of lengths is actually converging to the minimum length (i.e., the *global minimum*).

Let ESP denote the class of any Euclidean shortest path problem. An Euclidean path is a *δ-approximation (Euclidean) path* for an ESP problem iff its length is at most δ times the optimum solution.

We will also refer to a *convex hull algorithm* (see, e.g., [13] or Figure 13.7, [8]), which reads an ordered sequence of vertices of a planar simple polygonal curve ρ and outputs an ordered sequence of vertices of the convex hull of ρ; its running time is $\mathcal{O}(|V(\rho)|)$.

The paper is structured as follows: Section 2 provides in the first subsection approximation algorithms for the case of the fixed TPP, with either convex or not necessarily convex input polygons, and then in the second subsection some modifications of those two algorithms for solving the floating TPP, with either convex or not necessarily convex input polygons, in the approximate sense. Section 3 reports about experiments, and Section 4 concludes.

3 The Algorithms

In this section we partially answer the open problem mentioned in Section 2 by an approximation algorithm (Algorithm 2). However, we do not only deal with the fixed TPP, we also discuss an approximate solution for the floating TPP.

3.1 Approximation Algorithms for the Fixed TPP

First we provide an algorithm that only is guaranteed to find a fixed TPP solution as a *local minima*, and not necessarily as the intended global minima. However, if the input polygons P_i are all convex, then this simple algorithm already outputs an approximate fixed TPP solution (with adjustable accuracy) in the global sense.

Algorithm 1 (RBA for a sequence of pairwise disjoint simple polygons)
Input: A sequence of k pairwise disjoint simple polygons $P_1, P_2, \ldots, P_k$ in the same plane π; two points p, $q \notin \bigcup_{i=1}^{k} P_i$, and an accuracy constant $\varepsilon > 0$.
Output: A sequence $\langle p, p_1, p_2, \ldots, p_k, q\rangle$ which starts at $p = p_0$, then visits polygons P_i at points p_i in the given order, and finally ends at $q = p_{k+1}$.

1: For each $i \in \{1, 2, \ldots, k\}$, let initial vertex p_i be a vertex of P_i.
2: Let $L_0 = \infty$. Calculate $L_1 = \sum_{i=0}^{k} d_e(p_i, p_{i+1})$, where $p_0 = p$ and $p_{k+1} = q$.
3: **while** $L_0 - L_1 \geq \varepsilon$ **do**
4: **for** $i = 1, 2, \ldots, k$ **do**
5: Compute a point $q_i \in \partial P_i$ such that (see Fig. 4) $d_e(p_{i-1}, q_i) + d_e(q_i, p_{i+1}) = \min\{d_e(p_{i-1}, p) + d_e(p, p_{i+1}) : p \in \partial P_i\}$
6: Update the path $\langle p, p_1, p_2, \ldots, p_k, q\rangle$ by replacing p_i by q_i.
7: **end for**
8: Let $L_0 = L_1$ and calculate $L_1 = \sum_{i=0}^{k} d_e(p_i, p_{i+1})$.
9: **end while**
10: Return $\langle p, p_1, p_2, \ldots, p_k, q\rangle$.

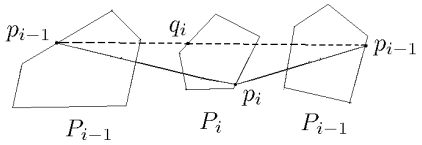

Figure 4: Initialization in Step 5 of Algorithm 1. Point p_i moves into a new position q_i.

This algorithm calculates an $(1 + 4k \times r(\varepsilon)/L)$-approximate solution for the fixed TPP and not necessarily only convex polygons P_i, where L is the length of a shortest path (i.e., the intended *global minimum*), $r(\varepsilon)$ the upper error bound for distances between p_i and its corresponding optimal vertex p'_i (i.e., $d_e(p_i, p'_i) \leq r(\varepsilon)$, for $i = 1, \ldots, k$, and d_e is the Euclidean distance). This is because for each $i \in \{1, 2, \ldots, k\}$, the error of the difference between $d_e(p_i, p_{i+1})$ and $d_e(p'_i, p'_{i+1}))$ is at most $4 \times r(\varepsilon)$ because of $d_e(p_i, p'_i) \leq r(\varepsilon)$. We obtain that

$$L \leq \sum_{i=0}^{k} d_e(p_i, p_{i+1}) \leq \sum_{i=0}^{k} [d_e(p'_i, p'_{i+1}) + 4 \times r(\varepsilon)] = L + 4k \times r(\varepsilon)$$

Thus, the output path is an $\{1 + 4k \times r(\varepsilon)/L\}$-approximation path.

Let $\kappa(\varepsilon) = \frac{L_0 - L}{\varepsilon}$ be a function which only depends upon the difference (!) between the lengths L_0 of an initial path and L of the optimum path, and the accuracy constant ε. Let L_m be the length of the m-th updated path, for $m = 0, 1, 2, \ldots$, with $L_m - L_{m+1} \geq \varepsilon$ (otherwise the algorithm stops). It follows that

$$\kappa(\varepsilon) = \frac{L_0 - L}{\varepsilon} \geq 1 + \frac{L_1 - L}{\varepsilon} \geq \cdots \geq m + \frac{L_m - L}{\varepsilon} \quad (1)$$

The sequence $\{m + \frac{L_m - L}{\varepsilon}\}$ is monotonously decreasing, lower bounded by 0, and stops at the first m_0 where $L_{m_0} - L_{m_0+1} < \varepsilon$. This defines a *local minimum* in this approximation process. However, it is still possible that L_{m_0+1} is not yet "close" to L in the case of nonconvex polygons. For convex input polygons, we may apply Lemma 1 in [2] which is as follows: For the unconstrained TPP, if all input polygons are pairwise disjoint and convex, then local optimality is equivalent to global optimality. We immediately obtain the following

Corollary 1 *If all input polygons are convex then Algorithm 1 outputs an approximate global solution for the fixed TPP.*

For the case of convex polygons, it is even possible to derive an explicit expression for the upper bound $r(\varepsilon)$. Obviously, $\lim_{\varepsilon \to 0} r(\varepsilon) = 0$. Thus, Algorithms 1 may be "tuned" by a very small $\varepsilon > 0$ to be of very high accuracy. The time complexity of the algorithm will be discussed later.

Now we provide a second (heuristic) algorithm which applies Algorithm 1 on the convex hulls $C(P)$ of the input polygons P in order to obtain an "improved" initial path whose vertices are located on the frontier of the convex hulls; then we transform this path in the algorithm into another one such that its vertices are on the frontier of the input polygons; finally, the algorithm applies the first algorithm on the input polygons to find a further improved solution to the fixed TPP.

Algorithm 2 (Algorithm for the fixed TPP; polygons may be nonconvex)
Input: A sequence of k simple polygons $P_1, P_2, \ldots, P_k$ such that the convex hulls $C(P_1), C(P_2), \ldots, C(P_k)$ are pairwise disjoint; two points p, $q \notin \bigcup_{i=1}^{k} C(P_i)$, and an accuracy constant $\varepsilon > 0$.
Output: A sequence $\langle p, p_1, p_2, \ldots, p_k, q \rangle$ which starts at p, then visits polygon P_i at p_i in the given order, and finally ends at q.

1: For $i \in \{1, 2, \ldots, k\}$, apply the Melkman algorithm for computing $C(P_i)$.
2: Let $C(P_1), C(P_2), \ldots, C(P_k)$, p, and q be the input of Algorithm 1 for computing an approximate shortest route $\langle p, p_1, \ldots, p_k, q \rangle$.
3: For $i = 1, 2, \ldots, k-1$, find a point $q_i \in \partial P_i$ such that $d_e(p_{i-1}, q_i) + d_e(q_i, p_{i+1}) = \min\{d_e(p_{i-1}, p) + d_e(p, p_{i+1}) : p \in \partial P_i\}$. Update the path for each i by $p_i = q_i$.
4: Let $P_1, P_2, \ldots, P_k$, p and q be the input of Algorithm 1, and points p_i as obtained in Step 3 are the initial vertices p_i in Step 1 of Algorithm 1. Continue with running Algorithm 1.
5: Return $\langle p, p_1, \ldots, p_{k-1}, p_k, q \rangle$ as provided in Step 4.

Step 2 iterates through the convex hulls. The iteration through step sets P_i only occurs in Step 4 (i.e., when applying Subalgorithm 1 for a second time, using the same ε). Algorithm 2 provides an $(1 + (L_2 - L_1)/L)$-approximate global solution for the floating TPP, where L is the length of an optimal path; L_1 is the length of the path obtained in Step 2; L_2 the length of the final path obtained in Step 5. Note that $L_2 \geq L_1$, and $L_2 = L_1$ if all polygons P_i are convex.

Theorem 1 *Algorithms 1 and 2 may be computed in time $\kappa(\varepsilon)\mathcal{O}(n)$, where n is the total number of vertices of the involved k polygons P_i.*

Proof. In Step 5 of Algorithm 1, each locally optimal point q_i can be computed in the order of $|V(P_i)|$ operations, where $V(P_i)$ is the set of vertices of P_i. Thus, each iteration of the for loop takes in the order of n operations at most. In theory, the number of runs through the outer while-loop is upper bounded by $\kappa(\varepsilon)$; see Equ. (1). Thus, Algorithm 1 will run in time $\kappa(\varepsilon)\mathcal{O}(n)$. □

Practically, extensive experiments showed that $\kappa(\varepsilon)$ was always much too large to estimate the actual number of runs through the outer while-loop. Obviously, $\kappa(\varepsilon)$ also depends on the selection of the initial path. By taking fixed initial points p_i (e.g., uppermost, leftmost vertex of P_i), the function $\kappa(\varepsilon)$ would only depend on ε and the configuration of input polygons P_i.

3.2 Approximation Algorithms for the Floating TPP

In this section, we propose two algorithms for the floating TPP. They are derived in a straightforward way from the two algorithms in the previous subsection.

Algorithm 3 (The "floating version" of Algorithm 1)
Input: A sequence of k pairwise disjoint simple polygons $P_0, P_1, \ldots, P_{k-1}$ in a plane π; an accuracy constant $\varepsilon > 0$.
Output: A sequence $\langle p_0, p_1, p_2, \ldots, p_k \rangle$ (where $p_k = p_0$) which visits polygon P_i at p_i in the given order $i = 0, 1, 2, \ldots, k-1$, and finally $i = 0$ again).

1: For each $i \in \{0, 1, \ldots, k-1\}$, let initial p_i be a vertex of P_i.
2: Let $L_0 = \infty$. Calculate $L_1 = \sum_{i=0}^{k-1} d_e(p_i, p_{i+1})$.
3: **while** $L_0 - L_1 \geq \varepsilon$ **do**
4: **for** $i = 0, 1, \ldots, k-1$ **do**
5: Compute a point $q_i \in \partial P_i$ such that $d_e(p_{i-1}, q_i) + d_e(q_i, p_{i+1}) = \min\{d_e(p_{i-1}, p) + d_e(p, p_{i+1}) : p \in \partial P_i\}$
6: Update the route $\langle p_0, p_1, \ldots, p_{k-1} \rangle$ by replacing p_i by q_i.
7: **end for**
8: Let L_0 be L_1 and calculate $L_1 = \sum_{i=0}^{k-1} d_e(p_i, p_{i+1})$.
9: **end while**
10: Return $\langle p_0, p_1, \ldots, p_{k-1}, p_k \rangle$.

This algorithm calculates an $[1 + 4k \cdot r(\varepsilon)/L]$-approximate solution for the floating TPP and convex polygons P_i, where L is the length of an optimal path, and $r(\varepsilon)$ the upper error bound for distances between p_i and a corresponding optimal vertex p'_i. Regarding a proof of correctness for Algorithm 3 and convex polygons P_i, it is analogous to the proof of Theorem 3 in [12]. (This proof is actually theoretically challenging, but to long for this conference paper.) The method of [2], as cited in the subsection before for the fixed TPP and convex polygons P_i, is not applicable here for showing correctness for the case of convex input polygons.

Algorithm 4 The "floating version" of Algorithm 2
Input: A sequence of k simple polygons $P_0, P_1, \ldots, P_{k-1}$ such that convex hulls $C(P_0), C(P_1), \ldots, C(P_{k-1})$ are pairwise disjoint and an accuracy constant $\varepsilon > 0$.
Output: A sequence $\langle p_0, p_1, p_2, \ldots, p_{k-1}, p_k \rangle$ (where $p_k = p_0$) which visits polygon P_i at p_i in the given order $i = 0, 1, 2, \ldots, k-1$, and then $i = 0$ again).

1: For $i \in \{0, 1, \ldots, k-1\}$, apply the Melkman algorithm for computing $C(P_i)$.
2: Let $C(P_0), C(P_1), \ldots, C(P_{k-1})$ be the input of Algorithm 3 for computing an approximate shortest route $\langle p_0, p_1, \ldots, p_k \rangle$.
3: For $i = 0, 1, 2, \ldots, k-1$, find a point $q_i \in \partial P_i$ such that $d_e(p_{i-1}, q_i) + d_e(q_i, p_{i+1}) = \min\{d_e(p_{i-1}, p) + d_e(p, p_{i+1}) : p \in \partial P_i\}$. Update p_i by letting p_i be q_i.
4: Let $P_0, P_1, \ldots, P_{k-1}$ be the input of Algorithm 3, and points p_i as obtained from Step 3 be the initial points in Step 1 of Algorithm 3 for computing an approximate shortest route $\langle p_0, p_1, \ldots, p_{k-1}, p_k \rangle$.
5: Return $\langle p_0, p_1, \ldots, p_{k-1}, p_k \rangle$.

This algorithm provides an $(1 + (L_2 - L_1)/L)$-approximate solution for the floating TPP with not necessarily only convex polygons P_i, where L is the length of an optimal path, L_1 is the length of the path obtained in Step 2, and L_2 the length of the final path in Step 5.

Regarding the time complexity of Algorithms 3 and 4, it is obvious that they are the same as that of Algorithms 1 and 2, that is in time $\kappa(\varepsilon)\mathcal{O}(n)$, where n is the total number of vertices of the k involved polygons P_i, and $\kappa(\varepsilon)$ is the function as defined before.

4 Experimental Results

Both Algorithms 2 and 4 were implemented in Java. On the left (the right) in Figure 5, the red route is obtained by Step 2 of Algorithm 2 (of Algorithm 4), the blue one is the initial route in Step 4 of Algorithm 2 (of Algorithm 4), and the green one is the final route of Algorithm 2 (of Algorithm 4) when applying $\varepsilon = 10^{-10}$. Routes follow a defined order of those polygons. In the

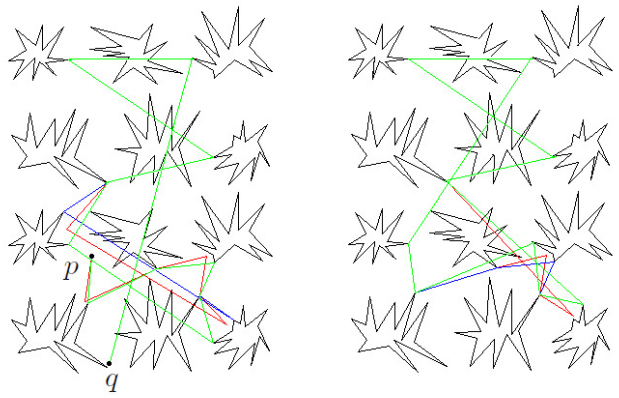

Figure 5: Routes as calculated in Steps 2 (red), 4 (blue), and 5 (green) of Algorithms 2 (left) and 4 (right).

table below, L_1, L_2, and L are as defined in the output of Algorithm 2, while L'_1, L'_2, and L' are the corresponding values defined in the output of Algorithm 4:

L_1	L_2	L	δ
2867·069	2888·999	2887·736	1·0076
L'_1	L'_2	L'	δ'
2521·294	2532·700	2532·700	1·000

Values δ and δ' are defined by $(1+(L_2-L_1)/L)$ or $(1+(L'_2-L'_1)/L')$, respectively. Note their closeness to 1·0! Both L and L' are approximate because they are found by running Algorithms 1 and 2 for 100,000 times (each time with a randomly selected initial path), and then selecting the minimum. Note that $L_2 \geq L$ ($L'_2 \geq L'$) follows from being approximate solutions for the fixed or the floating TPP. – For some further examples of measured run times, see Fig. 6.

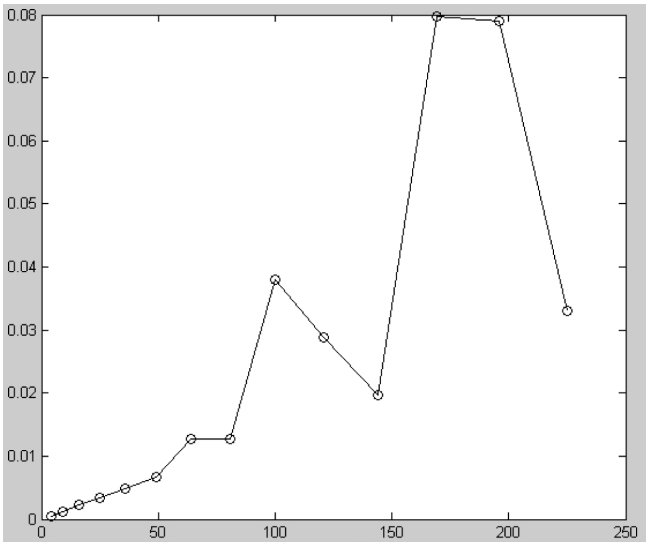

Figure 6: Running time (in seconds) of Algorithm 4, for numbers of randomly generated polygons up to 230. Implementation: in Java on a PC with Pentium Dual-Core CPU E5200 2·50 GHz, 1·99 GB memory; all input polygons with 14 vertices.

5 Concluding Remarks and Future Work

In this paper we present $(1 + (L_2 - L_1)/L)$-approximation algorithms for finding approximate solutions for both the fixed and the floating TPP, where simple input polygons have to satisfy the condition that their convex hulls are pairwise disjoint.

Our extensive experimental results indicate that the theoretical upper time bound $\kappa(\varepsilon)\mathcal{O}(n)$ may be replaced in practice by an upper bound $\mathcal{O}(n^2)$.

Our algorithms are suitable to solve both the fixed and floating TPP and have identical theoretical time complexity while the method of [2] is only suitable to the fixed TPP and convex polygons. Our method could be useful for solving the floating watchman route problem, the floating safari problem, and the floating zookeeper problem. Moreover, our algorithms are simpler and easier to understand and implement than that of [2].

In future, we will generalize our algorithms for handling cases where the convex hulls of the polygons are not necessarily pairwise disjoint.

References

[1] M. Dror. Polygon plate-cutting with a given order. *IIE Transactions*, **31**:271–274, 1999.

[2] M. Dror, A. Efrat, A. Lubiw, and J. Mitchell. Touring a sequence of polygons. In Proc. *STOC*, pages 473–482, 2003.

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

[4] A. M. Geoffrion. Lagrangian relaxation and its uses in integer programming. In *Mathematical Programming Study*, volume 2, pages 82–114. North Holland, Amsterdam, 1974.

[5] M. Guignard and S. Kim. Lagrangian decomposition: a model yielding stronger Lagrangian bounds. *Mathematical Programming*, **39**:215–228, 1987.

[6] D. S. Hochbaum (editor). *Approximation Algorithms for NP-Hard Problems*. PWS Pub. Co.,Boston, 1997.

[7] J. Hoeft and U. S. Palekar. Heuristics for the plate-cutting traveling salesman problem. *IIE Transactions*, **29**:719–731, 1997.

[8] R. Klette and A. Rosenfeld. *Digital Geometry*. Morgan Kaufmann, San Francisco, 2004.

[9] G. Laporte, H. Mercure, and Y. Nobert. Generalized traveling salesman problem through n clusters. *Discrete Applied Mathematics*, **18**:185–197, 1987.

[10] E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys. *The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization*. John Wiley and Sons, New York, 1985.

[11] F. Li and R. Klette. Rubberband algorithms for solving various 2D or 3D shortest path problems. Invited talk, in IEEE Proc. *Computing: Theory and Applications*, The Indian Statistical Institute, Kolkata, pages 9 - 18, 2007.

[12] F. Li and R. Klette. Watchman Route in a Simple Polygon with a Rubberband Algorithm (download source of Algorithm 1), MI-tech report-51. `www.mi.auckland.ac.nz/index.php?option=com_content&view=article&id=127&Itemid=113`

[13] A. Melkman. On-line construction of the convex hull of a simple polygon. *Information Processing Letters*, **25**:11–12, 1987.

[14] 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.

[15] C. E. Noon and J. C. Bean. An efficient transformation of the generalized traveling salesman problem. *INFOR*, **31**:39–44, 1993.