Touring Polygons: An Approximation Algorithm

Amirhossein Mozafari and Alireza Zarei

Department of Mathematical Sciences
Sharif University of Technology

Abstract. In this paper, we introduce a new version of the shortest path problem appeared in many applications. In this problem, there is a start point s, an end point t, an ordered sequence $\mathcal{S}=(S_0 = s, S_1, ..., S_k, S_{k+1} = t)$ of sets of polygons, and an ordered sequence $\mathcal{F}=(F_0, ..., F_k)$ of simple polygons named fences in $\Re^2$ such that each fence F_i contains polygons of S_i and S_{i+1}. The goal is to find a path of minimum possible length from s to t which orderly touches the sets of polygons of $\mathcal{S}$ in at least one point supporting the fences constraints. This is the general version of the previously answered Touring Polygons Problem (TPP). We prove that this problem is NP-Hard and propose a precision sensitive FPTAS algorithm of $O(k^2n^2/\epsilon^2)$ time complexity where n is the total complexity of polygons and fences.

Keywords: Computational geometry, approximation algorithm, touring polygons, minimum link path.

1 Introduction

Finding a shortest path is a basic subroutine in computational geometry and appears in many applications in mathematics and engineering. There are several types of shortest path problems. In its most conventional form, we have a weighted graph and the problem is to obtain the shortest path (path of minimum weight) from a source node to a destination[5]. In this paper, we introduce a special version of the shortest path problem in which we have an ordered set $\mathcal{S}=(S_0, ..., S_{k+1})$ of sets of polygons, a start point s, an end point t, and an ordered set of polygonal fences $\mathcal{F}=(F_0, ..., F_k)$ in $\Re^2$, such that F_i contains S_i and S_{i+1}. In this notation we assume that S_0 is the start point s and S_{k+1} is the end point t. The goal is to find the shortest path P (path of minimum length) from s to t such that P intersects at least one point of a polygon from each set S_i according to their order in such a way that the portion of this path from S_i to S_{i+1} lies inside F_i. This problem can be considered as a general version of the known Touring Polygons Problem (TPP)[6]. In TPP we have only one polygon in each set S_i. We denote the general version in which each S_i may contain more than one polygon by TMP (Touring Multiple-polygons Problem). Fig. 1 shows an example of TPP and Fig. 2 shows an example of TMP. In the unconstrained version of TMP and TPP, denoted by UTMP and UTPP respectively, all fences F_i are assumed to be the whole plane which means that there

S. Arumugam and B. Smyth (Eds.): IWOCA 2012, LNCS 7643, pp. 110–121, 2012.

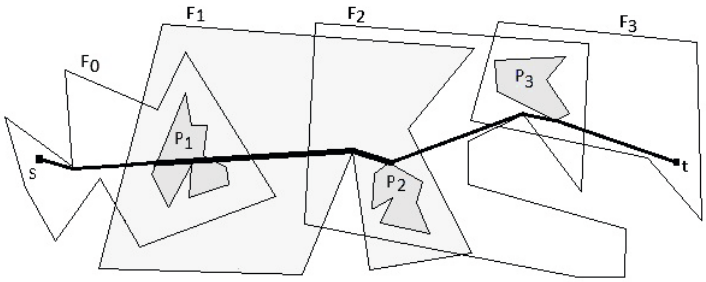

Fig. 1. An example of the TPP with polygons (P_1, P_2, P_3) and fences (F_0, F_1, F_2, F_3)

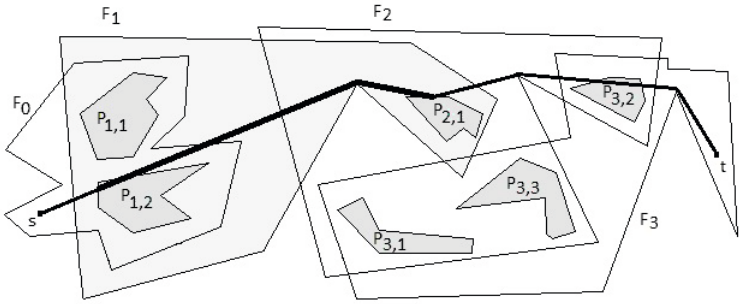

Fig. 2. The TMP with three sets of polygons $(\{P_{1,1}, P_{1,2}\}, \{P_{2,1}\}, \{P_{3,1}, P_{3,2}, P_{3,3}\})$ and fences (F_0, F_1, F_2, F_3)

is no constraint for the path from S_i to S_{i+1}. This problem has applications in several well-known problems in computational geometry including Watchman Route[4,11], Zookeeper[9], Safari[12], and Part Cutting[7] problems. Dror *et.al* [6] proved that TPP is NP-Hard when the polygons are non-convex and allowed to intersect each other. This implies that our problem is also NP-Hard when the polygons in $\mathcal{S}$ can intersect each other. In this paper, we prove that TMP is NP-Hard even if the polygons are disjoint and convex.

In this paper, we propose a precision sensitive ϵ-approximation algorithm for the TMP which is based on solving the shortest path problem on a graph built on the vertices of the polygons and some extra vertices put on the boundaries of these polygons. In the rest of this paper, in Section 2 we propose a precision sensitive FPTAS algorithm for TMP when the polygons are disjoint. In Section 3, we analyse the efficiency of this algorithm and show that the running time of this algorithm is the same as the running time of the best known approximation algorithm for TPP. In Section 4 we extend this algorithm to the overlapped situations. In Section 5, we prove the NP-Hardness of the TMP.

2 The FPTAS Algorithm

In this section we assume that consecutive sets of polygons S_i and S_{i+1} are disjoint from each other. This means that for all $P_r \in S_i$ and $P_q \in S_{i+1}$ we have $P_r \cap P_q = \emptyset$. Recall that we set S_0 as a set that consists of the single point s and S_{k+1} as a set that has the single point t. In Section 4, we extend this algorithm to the overlapped case.

To obtain an $\epsilon-$approximation solution for this problem we use the *pseudo approximation technique* (PAT) described in [2]. The sketch of this method is as follows. If X is the space of all solutions of a problem and $x^* \in X$ is an optimal solution, X is classified into subsets $X_R \subseteq X$ for different values of a real parameter $R \geq 0$ which is called the search radius. This classification must satisfy three properties: (1) if $R_1 \leq R_2$, then $X_{R_1} \subseteq X_{R_2}$, (2) there exists $R^* \geq 0$ such that $X_{R^*} = X$. (3) if $length(x^*) \leq R$ then $length(x^*_R) = length(x^*)$ where x^*_R is an optimal solution in X_R. Having these properties, by constructing a pseudo approximation algorithm for the search radius R and iteratively running it for different values of R, an accurate $\epsilon-$approximation algorithm is obtained. To be able to use this method the pseudo approximation algorithm must have this property that for each $R \geq 0$ and fixed $\epsilon \geq 0$

$$length(x^{apr}_R) \leq length(x^*_R) + \epsilon R,$$

where x^{apr}_R is the solution obtained by the pseudo approximation algorithm for the search radius R.

Now, we describe how to use PAT for solving the TMP. Denote by X the set of all solutions (acceptable paths) in our problem and use a search radius $R \geq 0$ to classify all solutions in X. We define X_R as the set of all solutions which are completely inside the disk of radius R with center s. Also, we denote x^* as an optimal solution (solution with minimum length) in X and x^*_R an optimal solution in X_R (note that if $X_R = \emptyset$ we set $length(x^*_R) = \infty$).

In order to use PAT, we need to check the three properties that the definition of the search radius must satisfy. It is simple to check that these properties are satisfied by the definition of our search radius and the classification method. Therefore, if we can obtain a pseudo approximation algorithm, we can use the PAT method to obtain an $\epsilon-$approximation algorithm.

For fixed $R > 0$, we restrict our problem to this disk, i.e., we remove all parts of polygons and fences which are outside this disk. In this restriction, an edge e of a polygon is replaced by $e \cap D_R(s)$ where $D_R(s)$ is the disk of radius R and center s. We put $\lceil 4k/\epsilon \rceil$ points on each edge of polygons and divide each edge into $\lceil 4k/\epsilon \rceil + 1$ fragments of equal length. The length of each fragment is at most $2R\epsilon/4k$ (we call these points as extra points). We build a directed weighted visibility graph, DVG, which its vertex set is the set of vertices of polygons and fences and the extra points. An edge $\overrightarrow{uv}$ exists in DVG if and only if these conditions holds:

1. u corresponds to a vertex or extra point of S_i or a vertex of fence F_i.
2. v corresponds to a vertex or extra point of S_{i+1} or a vertex of fence F_i.
3. The corresponding points of u and v are visible from each other with respect to fence F_i, i.e.,their connecting segment lies completely inside fence F_i.

The weight of an edge $\overrightarrow{uv}$ is set to be the distance between the corresponding points of its vertices. We run a shortest path algorithm like Dijkstra[5] from s to t in this directed graph to obtain the path x_R^{apr}.

Lemma 1. If x_R^{apr} exists, it belongs to X_R.

Proof. According to its construction, all vertices of DVG lie inside $D_R(s)$. Therefore, the path x_R^{apr} entirely lies inside this disk. To complete the proof, We must show that x_R^{apr} is an acceptable path, i.e., it starts from s, ends at t, and intersects at least one polygon in each S_i in their correct order and satisfies the fences constraints. Trivially, x_R^{apr} starts from s and ends at t. Assume that $< p_r, p_{r+1}, ..., p_q >$ is a sub-path in x_R^{apr} where p_r and p_q are respectively the first vertices of x_R^{apr} belonging to the polygon sets S_i and S_{i+1} for $0 \leq i \leq k$. According to the direction of the edges in DVG, the outward edges from p_r is only to F_i and S_{i+1} vertices, and the inward edges to p_q is restricted to the vertices of F_i and S_i. Therefore, the vertices between p_{r+1} and p_{q-1} in path $< p_r, p_{r+1}, ..., p_q >$ only belong to the vertices of F_i (note that it is possible that p_r and p_q are directly connected by an edge which means that the path $< p_r, ..., p_q >$ is a single edge). Moreover, each edge $\overrightarrow{uv}$ which its start vertex belongs to S_i or F_i and its end vertex belongs to F_i or $Si + 1$ follows the visibility constraints of fence F_i. This implies that the sub-path $< p_r, ..., p_q >$ lies inside the fence F_i. Finally, we prove by contradiction that path X_R^{apr} intersects the polygon sets S_i according to their order. Assume that i is the smallest value which S_i polygons are not intersected by x_R^{apr} just after entering a polygon in S_{i-1}. According to the direction of the edges in DVG, after entering a polygon in S_{i-1} the path x_R^{apr} can only enter a vertex of F_{i-1} or S_i and to leave the vertices of F_{i-1} it must enter a vertex of S_i. Therefore, if x_R^{apr} ends at t, it must pass through a vertex of S_i after leaving S_{i-1}. □

Now, we analyse the relation between x_R^{apr} and x_R^*. We can locate a sequence of p_i points on x_R^* where p_i is the first intersection point of x_R^* and the set of polygons S_i after visiting the polygon sets $S_1, ..., S_{i-1}$. Here, p_0 and p_{k+1} are respectively the start point s and the end point t. Therefore, the optimal path x_R^* can be divided into k sub-paths which the i'th sub-path ($0 \leq i \leq k$) starts from a point $p_i \in S_i$ and ends at a point $p_{i+1} \in S_{i+1}$. These sub-paths are denoted by $x_R^*(i)$ (Fig. 3).

Each point p_i lies on the boundary of a polygon in S_i. This boundary point may be a vertex of a polygon in S_i, an extra point on an edge or a point on a fragment of length at most $R\epsilon/2k$. According to the definition of p_i points, each sub-path $x_R^*(i)$ lies inside fence F_i. Moreover, each sub-path $x_R^*(i)$ lies inside a geometric structure called *hourglass* defined as bellow.

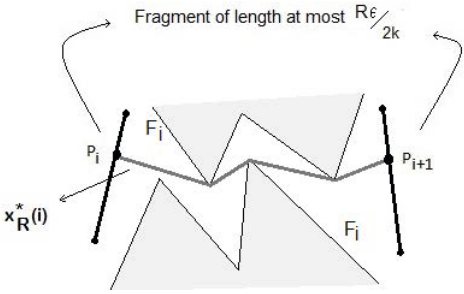

Fig. 3. Sub-path $x^*_R(i)$

Assume that b_s and b_e are respectively the fragments of length at most $R\epsilon/2k$ containing the points p_i and p_{i+1} of the sub-path $x^*_R(i)$. Note that in some cases b_s or b_e may be a single point. The corresponding hourglass of $x^*_R(i)$ is the region defined by these segments and the two shortest paths connecting the endpoints of b_s and b_e that $x^*_R(i)$ lies between them. Fig. 4 shows some configurations for hourglass shapes.

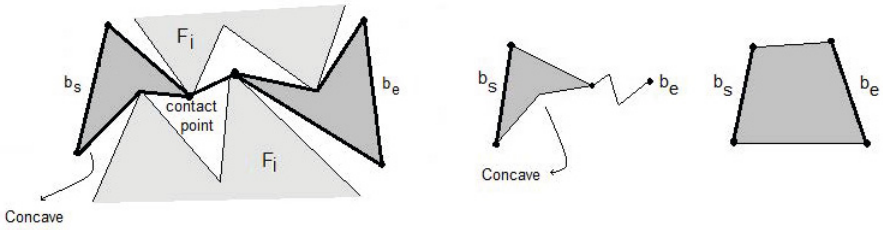

Fig. 4. Some configurations of hourglass shapes

We denote the sequence of hourglasses by $H_0, ..., H_k$ where each H_i contains $x^*_R(i)$. The end points of $x^*_R(i)$ lie on two edges of H_i that their lengths are at most $R\epsilon/2k$. These edges are shown as thick segments in Fig. 4 and we call them as the *base* edges of hourglass H_i. Let s_i and l_i be respectively the minimum and maximum length shortest paths that connect a point from one base edge of H_i to a point on the other base edge which completely lie inside H_i (or on its boundary). Fig. 5 shows some configurations for s_i and l_i paths. We define L_{min} and L_{max} as follows :

$$L_{min} = \sum_{i=0}^{k} |s_i|$$

$$L_{max} = \sum_{i=0}^{k} |l_i|$$

Lemma 2. $L_{max} \leq L_{min} + \epsilon R$.

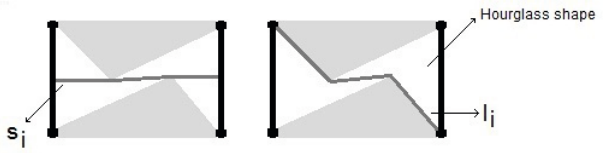

Fig. 5. Some configurations for s_i and l_i

Proof. To prove the relation between L_{min} and L_{max} we first obtain the relation between $|s_i|$ and $|l_i|$ and extend it to L_{min} and L_{max}. Assume that for a hourglass H_i, the paths s_i and l_i are respectively composed of the sequence of points $< p_{l_1}, ..., p_{l_r} >$ and $< p_{s_1}, .., p_{s_q} >$. The points p_{l_1} and p_{s_1} lie on one base edge of H_i and p_{l_r} and p_{s_q} lie on the other base edge. Consider a new path $m_i =< p_{l_1}, p_{s_1}, p_{s_2}, ..., p_{s_q}, p_{l_r} >$. The path m_i connects the end points of l_i and completely lies inside H_i. While l_i is the shortest path between these points, $|l_i| \leq |m_i|$. On the other hand, the length of the base edges on which the segment $p_{l_1}p_{s_1}$ and $p_{s_q}p_{l_r}$ lie are at most $R\epsilon/2k$. Therefore,

$$|l_i| \leq |m_i| \leq |s_i| + 2(R\epsilon/2k) = |s_i| + R\epsilon/k.$$

From the above relation we prove the relation between L_{min} and L_{max} :

$$L_{max} = \sum_{i=0}^{k} |l_i| \leq \sum_{i=0}^{k} |s_i| + k(R\epsilon/k) = L_{min} + \epsilon R. \qquad \square$$

Now, we can prove the relation between x_R^{apr} and x_R^* which is required in the pseudo approximation algorithm of the PAT method.

Lemma 3. $length(x_R^{apr}) \leq length(x_R^*) + \epsilon R$.

Proof. It is trivial that $|s_i| \leq |x_R^*(i)| \leq |l_i|$. Therefore, we have :

$$L_{min} \leq length(x_R^*) \leq L_{max}.$$

Moreover, $length(x_R^*) \leq length(x_R^{apr})$ and $length(x_R^{apr}) \leq L_{max}$. The reason of the latter inequality is that if we use a path from s to t which lies only on the boundary of the sequence of H_i regions, its length is at most L_{max} and it is a valid path in DVG. Therefore, the length of x_R^{apr} cannot be greater than the length of this path. Hence, we have :

$$L_{min} \leq length(x_R^*) \leq length(x_R^{apr}) \leq L_{max}.$$

Combining this relation with the result of Lemma 2 we obtain the final result:

$$L_{min} \leq length(x_R^*) \leq length(x_R^{apr}) \leq L_{max} \leq L_{min} + \epsilon R \leq length(x_R^*) + \epsilon R.$$

$$\Longrightarrow length(x_R^*) \leq length(x_R^{apr}) \leq length(x_R^*) + \epsilon R. \qquad \square$$

Now, we have all of the requirements of PAT and we can use this method to have the correct FPTAS algorithm. We assume that all inputs are rational numbers. If we set R^* as 2^L where L is the maximum bit length of the input integers, we can use the conversion procedure of PAT to obtain a *precision sensitive* $\epsilon-$approximation algorithm. In the next section we analyse the efficiency of this algorithm.

3 Efficiency of the Algorithm

The running time of this algorithm depends on the size of the built graph and running time of finding the shortest path from s to t in this graph. We first obtain the complexity of computing a pseudo approximation path for fixed ϵ and R. We have $O(k/\epsilon)$ points on each edge and if n is the complexity of our problem (number of vertices of all polygons and fences) we have $O(nk/\epsilon)$ vertices in the graph. Let f_i $(0 \leq i \leq k)$ be the number of these vertices inside F_i (vertices of F_i and vertices and extra points of S_i and S_{i+1}). We can construct visibility graph for each F_i in f_i^2 time[1]. The sum of f_is is $O(nk/\epsilon)$ so we can construct entire visibility graph in $O(n^2k^2/\epsilon^2)$ time. The number of vertices of this graph is $O(nk/\epsilon)$. Therefore, running Dijkstra algorithm on this graph takes $O(n^2k^2/\epsilon^2)$ time. Hence, we can obtain a pseudo approximation path in $O(n^2k^2/\epsilon^2)$ time. To obtain the ϵ-approximation path with the geometric search of the PAT method[2], we use the pseudo approximation algorithm $O(\log\log(R^*/length(x^*)))$ times. We assume that all inputs are rational numbers each of which consists of integer numerator and denominator. If we set R^* as 2^L where L is the maximum bit length of the input integer in our system, the maximum value of R^* is 2^L and minimum value of $length(x^*)$ is 2^{-L}. Then we need to run the pseudo approximation algorithm for $O(\log\log(2^{2L}))$ times. While on a typical machine L is constant, we must run the pseudo approximation algorithm a constant number of times. This means that the total running time of this algorithm is $O(k^2n^2/\epsilon^2)$.

4 Extending to the Overlapped Cases

In Section 2, we proposed an ϵ-approximation algorithm for solving the TMP when polygons in S_i are disjoint from polygons in S_{i+1}. In this section, we extend this algorithm to the cases where polygons in S_i are allowed to intersect polygons in S_{i+1}. Fig. 6 shows an example where the approximation factor of our algorithm is not depend on the value of ϵ and for arbitrarily small value of ϵ it remains large. In this example, we have three sets of polygons $(S_1 = \{P_{1,1}\}, S_2 = \{P_{2,1}\}, S_3 = \{P_{3,1}\})$ each of which has one polygon. For this configuration, the approximation factor of the algorithm is approximately 2 even for infinitely small value of ϵ.

This problem happens because in our algorithm the touring path is forced to touch the polygons in their boundaries. But, as seen in this example, we can obtain better approximation by touching some polygons ($P_{2,1}$ in this example)

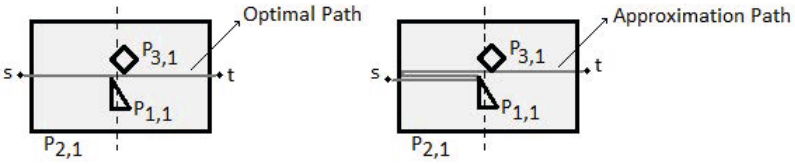

Fig. 6. A negative example for the first algorithm on intersecting polygons

in its interior. In order to solve problem we need to built DVG in such a way that handles such situations. We built DVG as follows. The vertex set of DVG is the same as before, but, there is a directed edge $\overrightarrow{uv}$ in this graph if and only if any one of the following conditions holds:

1. u and v are visible vertices of a fence with respect to that fence.
2. u is a vertex or an extra point on the boundary of a polygon of S_i which lies inside at least one polygon from each set $S_{i+1}, ..., S_{i+j}$ and does not lie inside a polygon of S_{i+j+1}, and v is a point of some polygon in S_{i+r} for $1 \leq r \leq j+1$ which u and v are visible from each other with respect to F_{i+r-1}, or,
   v is a vertex of F_{i+r} for $0 \leq r \leq j$ which is visible from u with respect to this fence.
3. u is a vertex of fence F_i and v is a vertex or extra point of a polygon of S_{i+1} and u and v are visible from each other with respect to F_i.

Lemma 4. Running Dijkstra algorithm from s to t on this graph returns a valid pseudo approximation path.

Proof. (Sketch) Let x_R^{apr} be this path. By the same argument as Lemma 1 and according to the construction of the graph, it is simple to show that each path from s to t in this graph touches all polygon sets in correct order supporting the fences constraints. This means that x_R^{apr} is a valid touring path.
To satisfy the pseudo approximation path requirement, we must show that $length(x_R^{apr}) \leq length(x_R^*) + \epsilon R$. Consider x_R^* as a sequence of $k+1$ sub-paths $(0 \leq i \leq k)$ such that $x_R^*(i)$ starts from the first point of x_R^* that lies on S_i after touring S_{i-1} and ends at the first point of x_R^* that lies on S_{i+1}. While S_i polygons may have intersections, a sub-path $x_R^*(i)$ may have zero length. It is simple to prove that the start point (end point) of $x_R^*(i)$ lies on the boundary of a polygon of a set S_j (S_l) $0 \leq j \leq i$ $(j \leq l \leq k+1)$. We denote the fragments that contain the start and end points of $x_R^*(i)$ by f_i^s and f_i^e, respectively. Trivially, $f_0^s = s$ and $f_{k+1}^e = t$. Moreover, $f_{i+1}^s = f_i^e$. We build a path P from s to t in DVG where $length(P) \leq length(x_R^*) + \epsilon R$. This path follows x_R^* in such a way that for each sub-path $x_R^*(i)$ which ends at a point on f_i^e, its corresponding sub-path in P, denoted by P_i, either (Case 1) ends at an endpoint of f_i^e or (Case 2) another fragment $g_i^e \in S_{i+1}$ which intersects f_i^e. Case 1 happens when the segment $x_i a$ lies inside F_{i+1} where x_i is the endpoint of $x_R^*(i)$ on f_i^e and a

is an endpoint of f_i^e at which P_i ends. Otherwise, Case 2 happens in where g_i^e is a fragment of S_{i+1} which its intersection point with f_i^e is the closest to x_i. It is simple to show that if $f_i^e \in S_{i+1}$ Case 1 happens and in Case 2 the fragment g_i^e always exists which intersects f_i^e.

Now, we inductively on i for $0 \leq i \leq k$ follow the path x_R^* and build the path P and show that in each step $0 \leq i \leq k$, $length(P)$ increases by at most $length(x_R^*(i)) + R\epsilon/k$. For $i = 0$, we start from s and if $s \in S_1$ then $x_R^*(0)$ and hence P_0 have zero length. Otherwise, $x_R^*(0)$ lies inside a hourglass with s and $f_0^e \in S_1$ as its bases. By the same argument as for the non-overlapping polygons, we can build P_0 from s to an endpoint of f_0^e with length at most $length(x_R^*(0)) + R\epsilon/k$. For $i > 0$, we consider two cases, $length(x_R^*(i)) = 0$ and $length(x_R^*(i)) \neq 0$ separately.

Assume that $length(x_R^*(i)) = 0$. Based on the cases applied on P_{i-1} and P_i, four options may happens which are shown in Fig. 7.

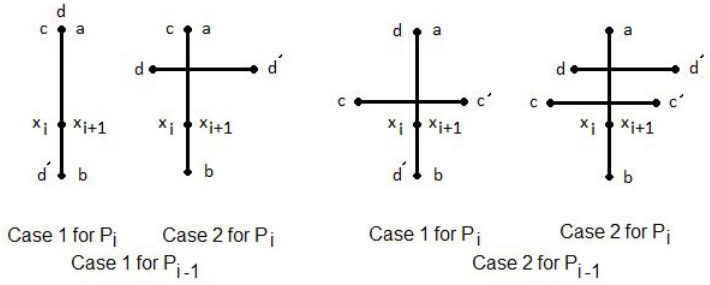

Fig. 7. Building P_i when $length(x_R^*(i)) = 0$

In this figure $f_i^s = f_i^e = f_{i-1}^e = ab$, x_i is the end point of $x_R^*(i)$, c is the endpoint of P_{i-1} and d is the endpoint of P_i. In all options we can find the union of paths P_{i-1} and P_i which starts from a and ends at an endpoint of dd' in such a way that their total length is at most $length(x_R^*(i-1)) + length(x_R^*(i)) + 2R\epsilon/k$. Now assume that $length(x_R^*(i)) \neq 0$. Hence, we also have four options based on the cases applied on P_{i-1} and P_i shown in Fig. 8 and we can build proper paths as well. □

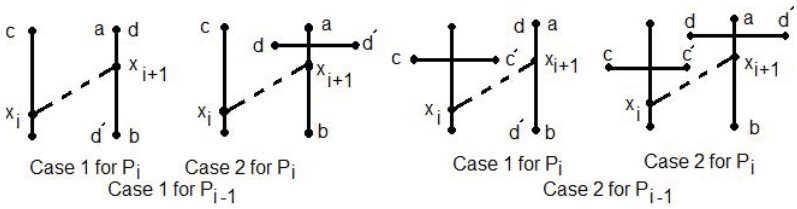

Fig. 8. Building P_i when $length(x_R^*(i)) \neq 0$

5 Complexity of the Problem

In this section, we show that UTMP (Unconstrained Touring Multiple-polygons Problem) is NP-Hard even for the L_1 norm and the case that all polygons are convex and disjoint from each other. Our proof is similar to the NP-Hardness proof of UTPP in Section 6 of [6] which itself is based on the NP-Hardness proof of three-dimensional shortest path problem[3]. This proof is a reduction from 3-SAT. Suppose that we have an instance of the 3-SAT problem with n variables $b_1, ..., b_n$ and m clauses $C_i = (l_{i1} \vee l_{i2} \vee l_{i3})$. For fixed points s and t we construct a sequence of sets of polygons with total complexity $O(n + m)$ for which solving the UTMP enables us to determine whether our 3-SAT problem has a satisfying truth assignment. We construct five kinds of gadgets : 2-way path splitter that doubles the number of shortest paths, 3-way path splitter that triples the number of shortest paths without changing their order (Fig. 9), path shuffler that performs a perfect shuffle of the input paths, literal filter that selects paths whose encodings have 0 or 1 in the i^{th} bit (literal filters consisting of n shufflers and one horizontal segment to stretch all paths having special bit equal to 0 or 1) (Fig. 10), clause filter that determines whether a specific clause has satisfying truth assignment (Fig 11).

In these gadgets we have only line segment polygons which are convex and disjoint from each other and their angle with the x-axis is $0, \pm 45$ and 90. In this construction, we use n 2-way splitters to create 2^n distinct paths each of which encodes a truth assignment for n variables. Then, we use sequence of m clause filters each consisting of three literal filters contained between two 3-way splitters. This permits us to filter those paths fail to satisfy each clause. Fig. 12 shows how we can select polygonal sets to build three parallel literal filters inside a clause filter. We need to put a blocker set after each shuffler. Each blocker set has a segment that determines whether the output paths of the shuffler need to be stretched. As shown in Fig. 10, if this segment is full no output path is forced to be stretched.

Note that we can always modify size and position of these gadgets to enforce that each input path directly goes to one of the segments without bending or intersecting endpoint of segments. Finally, we use 2-way splitters to collect all paths back to a single path that terminates at t. In this construction, all segments are in the plane and disjoint from each other. So, the initial 3-SAT problem has

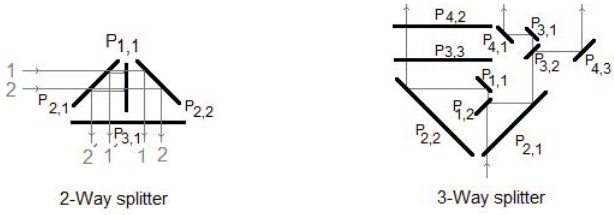

Fig. 9. Two and Three way path splitter

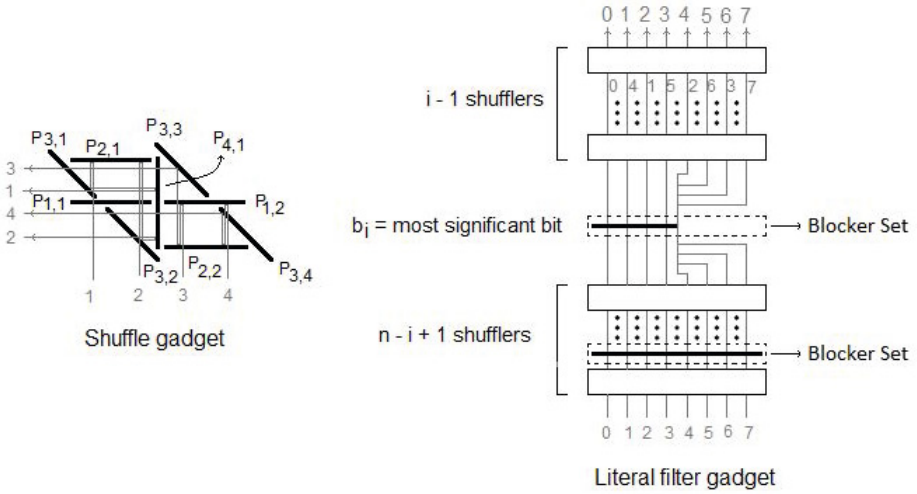

Fig. 10. Shuffle and Literal filter

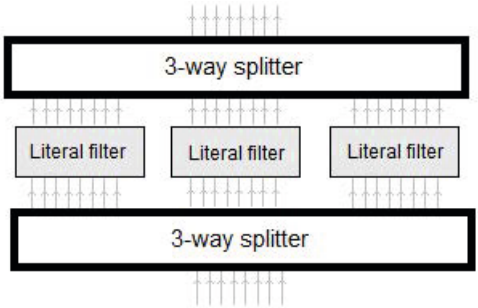

Fig. 11. Clause filter gadget

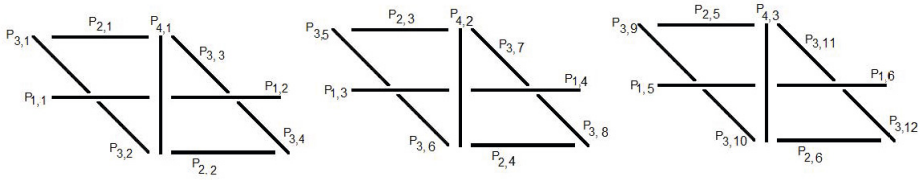

Fig. 12. Three shuffle gadgets in parallel

a satisfying truth assignment if and only if the solution of this UTMP problem is equal to the distance from s to t.

References

1. Asano, T., Asano, T., Guibas, L., Hershberger, J., Imai, H.: Visibility-polygon search and Euclidean shortest paths. In: Proc. 26th IEEE Symposium on Foundations of Computer Science, pp. 155–164 (1985)
2. Asano, T., Kirkpatrick, D., Yap, C.: Pseudo approximation algorithms, with applications to optimal motion planning. In: Proc. 18th Annu. ACM Sympos. Comput. Geom., pp. 170–178 (2002)
3. Canny, J., Reif, J.H.: New lower bound techniques for robot motion planning problems. In: Proc. 28th Annu. IEEE Sympos. Found. Comput. Sci, pp. 49–60 (1987)
4. Chin, W., Ntafos, S.: Shortest Watchman Routes in Simple Polygons. Discrete and Computational Geometry 6(1), 9–31 (1991)
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT press (2009) ISBN 978-0-262-03384-8
6. Dror, M., Efrat, A., Lubiw, A., Mitchell, J.: Touring a sequence of polygons. In: Proc. STOC 2003, pp. 473–482 (2003)
7. Dror, M.: Polygon plate-cutting with a given order. IIE Transactions 31, 271–274 (1999)
8. Guibas, L.J., Hershberger, J.: optimal shortest path queries in simple polygon. J. Comput. Syst. Sci. 39, 126–152 (1989)
9. Hershberger, J., Snoeyink, J.: An efficient solution to the zookeeper's problem. In: Proc. 6th Canadian Conf. on Comp. Geometry, pp. 104–109 (1994)
10. Li, F., Klette, R.: Rubberband algorithms for solving various 2D or 3D shortest path problems. In: Proc. Computing: Theory and Applications, The Indian Statistical Institute, Kolkata, pp. 9–18. IEEE (2007)
11. Tan, X., Hirata, T.: Constructing Shortest Watchman Routes by Divide and Conquer. In: Ng, K.W., Balasubramanian, N.V., Raghavan, P., Chin, F.Y.L. (eds.) ISAAC 1993. LNCS, vol. 762, pp. 68–77. Springer, Heidelberg (1993)
12. Tan, X., Hirata, T.: Shortest Safari Routes in Simple Polygons. In: Du, D.-Z., Zhang, X.-S. (eds.) ISAAC 1994. LNCS, vol. 834, pp. 523–531. Springer, Heidelberg (1994)