Path Planning with Objectives Minimum Length and Maximum Clearance

Mansoor Davoodi, Arman Rouhani, Maryam Sanisales

Path Planning with Objectives Minimum Length and Maximum Clearance

Mansoor Davoodi[0000−0003−1010−4121], Arman Rouhani(✉)[0000−0003−4822−484X], and Maryam Sanisales[0000−0002−3375−5085]

Department of Computer Science and Information Technology,
Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan, Iran
{mdmonfared,arman.rouhani,maryamsan}@iasbs.ac.ir

Abstract. In this paper, we study the problem of bi-objective path planning with the objectives minimizing the length and maximizing the *clearance* of the path, that is, maximizing the minimum distance between the path and the obstacles. The goal is to find Pareto optimal paths. We consider the case that the first objective is measured using the Manhattan metric and the second one using the Euclidean metric, and propose an $O(n^3 \log n)$ time algorithm, where n is the total number of vertices of the obstacles. Also, we state that the algorithm results in a $(\sqrt{2}, 1)$-approximation solution when both the objectives are measured using the Euclidean metric.

Keywords: Path planning · Shortest path · Bi-objective optimization · Clearance · Pareto optimality · Approximation algorithm

1 Introduction

Path Planning (PP) is one of the challenging problems in the field of robotics. The goal is to find the optimal path(s) for two given start and destination points among a set of obstacles. However, minimizing the length of the path is usually considered as the optimality criterion. The application of the other objectives of the problem, such as smoothness and clearance has been also considered in the literature [3]. For example, in many applications, the robot needs to move around in order to perform its task properly. The need for moving around the environment, led to the question of what path a robot can take to accomplish its task, in addition to being safe. In this paper, we define the *optimal path* with respect to two objectives minimizing the length of the path and maximizing the clearance (i.e., the minimum distance between the path and the obstacles).

Computing the Visibility Graph (VG) of the obstacles is a classical approach to obtain the path with the minimum length. For a set of polygonal obstacles with n vertices, VG is computed in $O(n^2 \log n)$ time using a ray shooting technique [6]. VG is one of the best-known approaches to obtain the shortest path where the distance between the path and the obstacles is equal to zero – a path with clearance zero. Hershberger et al. [7] proposed an efficient planar structure for the PP problem in $O(n \log n)$ time.

Clarkson et al. [1] presented two algorithms for finding the rectilinear shortest path amongst a set of non-intersecting simple polygonal obstacles in the plane. One of the algorithms takes $O(n \log^2 n)$ time and the other one requires $O(n \log^{3/2} n)$ time. Likewise, Mitchel et al. [10] used a continuous Dijkstra's algorithm technique which considers the propagation of a "wavefront" for computing the L_1 shortest path among polygonal obstacles in the plane and runs in $O(|E| \log n)$ time and $O(|E|)$ space, where $|E| = O(n \log n)$ is the number of "events". Inkulu et al. [9] presented an $O(n + m(\log n)^{3/2})$ time algorithm to find an $L1$ shortest path, where m is the number of non-intersecting simple polygonal obstacles.

Wein et al. [11] introduced a new type of visibility structure – called *Visibility-Voronoi Diagram* – by using a combination of the Voronoi diagram and VG to find the shortest path for a predefined value λ of clearance. They considered the PP problem in the setting of single objective optimization, that is, minimizing the length subject to minimum clearance λ. Geraerts [5] proposed a new data structure – called *Explicit Road Map* – that creates the shortest possible path with the maximum possible clearance. The introduced structure is useful for computing the path in the corridor spaces. Davoodi [2] studied the problem of bi-objective PP in a grid workspace with objectives minimizing the length of the path and maximizing its clearance. He also studied the problem in the continuous space under the Manhattan metric, and proposed an $O(n^3)$ time algorithm for a set of n vertical segments as the obstacles.

We study the problem of bi-objective PP with the objectives minimizing the length of the path and maximizing its *clearance*, that is, the minimum distance between the path and the obstacles. The goal is computing Pareto optimal solutions, that is, the paths which cannot be shortened if and only if their clearance is minimized. Since this problem is a bi-objective optimization problem in a continuous workspace, there is an infinite number of Pareto optimal solutions. So, it is impossible to provide a polynomial algorithm to compute all the Pareto optimal solutions. To bypass this issue, we focus on different Pareto optimal solutions, the paths with different middle points. We consider a PP workspace with a set of polygonal obstacles with total n vertices; and propose an $O(n^3 \log n)$ time algorithm to compute the Pareto optimal solutions where the length of the paths is measured using the Manhattan and the clearance is measured using the Euclidean metric. Also, we show the algorithm is an efficient approximation approach when both objectives are measured using the Euclidean metric. The problem studied in [2] discusses a special instance of the problem studied here which proposes an algorithm for a set of vertical line segments as the obstacles. Here we study the problem for a set of polygonal obstacles as the obstacles that is a more general case.

This paper is organized in five sections. In Section 2, we formally define the problem as bi-objective PP and give some preliminaries. In Section 3, an algorithm is proposed for the problem, where the length and clearance objectives are measured using the Manhattan and Euclidean metrics, respectively. The complexity analysis is also in this section. Section 4, extends the results to the

case which both objectives are measured using the Euclidean metric and provides an approximation algorithm. Finally in Section 5, future work and conclusion are presented.

2 Preliminaries

In this section, we provide some notations to formally define the bi-objective path planning (PP) problem. Let $\mathcal{P} = \{P_1, P_2, ..., P_m\}$ be a set of simple non-intersecting polygonal obstacles, and s and t be the start and destination points in the plane. We denote the set of all vertices of the obstacles by $\mathcal{V}$ and let $n = |\mathcal{V}|$. Also, we denote any collision-free path starting from s and ending at t with the notation *s-t-path*. Let $\pi = < s = v_0, v_1, ..., v_k, v_{k+1} = t >$ be a rectilinear *s-t*-path containing k *breakpoints* $v_1, v_2, ..., v_k$, and $\lambda \geqslant 0$ be a favored distance we want to keep away from the obstacles as *clearance*. We define $\mathrm{L}(\pi)$ to be the length of π under the Manhattan metric, and $\mathrm{C}(\pi) = \lambda$ as the clearance of π under the Euclidean metric, i.e., the minimum distance from the obstacles along π. In the problem of bi-objective PP, we wish to minimize $\mathrm{L}(\pi)$ and maximize $\mathrm{C}(\pi)$.

Let π_1 and π_2 be two *s-t*-paths. We say that π_1 *dominates* π_2, if π_1 is better than π_2 in both objectives. Precisely, if $L(\pi_1) < L(\pi_2)$ and $C(\pi_1) \geq C(\pi_2)$, or $L(\pi_1) \leq L(\pi_2)$ and $C(\pi_1) > C(\pi_2)$. We also say π_1 and π_2 are *non-dominated* paths if none of them dominates the other one. *Pareto optimal paths* which are denoted by Π^*, are the set of all non-dominated paths in the workspace.

Since Π^* is an infinite set in the bi-objective PP problem, computing and reporting all the Pareto optimal path is impossible. Thus, to have a well-defined problem, we focus on finding the *extreme* Pareto optimal path defined as follows. Let π_1 be the Pareto optimal path with $C(\pi_1) = \lambda = 0$. By continuously increasing λ, different Pareto optimal paths can be obtained using continuously moving the breakpoints of π_1 in the workspace. This process is possible until λ meets a *critical* value λ_0. See Fig. 1 to obtain the path π_2 by continuously moving the breakpoints of π_1. In this step, to find a Pareto optimal path π_3 with $C(\pi_3) > \lambda_0$, it is necessary that some breakpoints of π_2 are changed totally (or say jumped) , that means, π_3 cannot be obtained using a continuous moving of the breakpoints of π_2 any more. We call π_1 and π_2 as the *extreme Pareto optimal paths*. For example, the subfigures (a)-(d) in Fig. 1 show four extreme Pareto optimal paths π_1–π_4, respectively.

A different point of view for the definition of the extreme Pareto optimal path comes from the *objective space*. Since any *s-t*-path π has two objective values $L(\pi)$ and $C(\pi)$, so, it can be mapped to the 2-dimensional objective space $L(\pi)$ and $C(\pi)$. The image of Π^* in the objective space is called *Pareto fronts*. See Fig. 1 (e). The extreme Pareto optimal path are the paths corresponding with the endpoints of Pareto fronts. Indeed, the projection of the Pareto fronts on the axis $C(\pi)$ is a continuous interval $[0, \lambda_{max}]$, where λ_{max} is the maximum possible clearance in the workspace. However, the projection of Pareto fronts on

the axis $L(\pi)$ is a set of non-intersected intervals. The lower and upper bound of these intervals correspond with extreme Pareto optimal paths in the workspace.

In the following, we discuss the relationship between the defined bi-objective PP problem and the classic version of the PP problem which is known as shortest path planning problem.

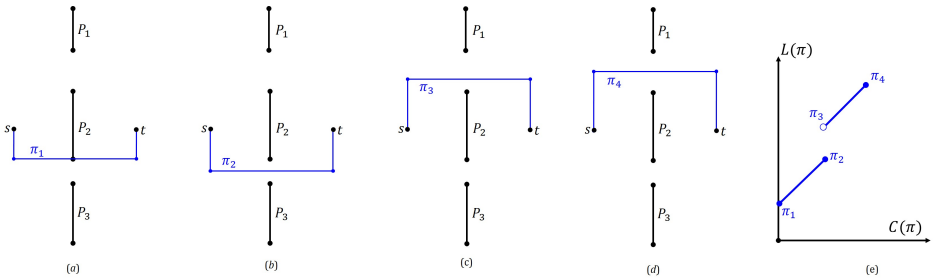


Fig. 1: An example of bi-objective path planning workspace ((a)–(d)) and its objective space (e). The Pareto optimal paths π_1 and π_2 lie on the same Pareto front, and the Pareto optimal paths π_3 and π_4 lie on another Pareto front. Indeed, the path π_2 can be obtained by continuously moving the breakpoints of π_1 by increasing λ, however, the path π_3 cannot be obtained.

In the classic single-objective version of the problem, the goal is to minimize $\mathrm{L}(\pi)$. Therefore, only one path is required as the output. As mentioned earlier, there are several studies that solve the problem of finding the rectilinear shortest path among polygonal obstacles. Although this version is single-objective, it is the same as our problem when $\mathrm{C}(\pi) = \lambda = 0$. Precisely, one can solve the bi-objective path planning problem by considering it as many shortest path problems at each desired clearance λ. Instead of keeping a distance λ from obstacles, we can grow (or say fatten) the obstacles with size λ. If we are given a polygon P, the Minkowski sum of P and a disk $\mathcal{D}(\lambda)$ of radius λ is the set of all points whose distance from P is less than λ.

Let $\mathcal{P}(\lambda) = \{P_1(\lambda), P_2(\lambda), ..., P_m(\lambda)\}$ be the set of fattened polygons with $\mathcal{D}(\lambda)$. It is obvious that the breakpoints of the shortest s-t-path for a given clearance λ belong to $\mathcal{P}(\lambda)$ [2]. Therefore, we can increase λ slightly from 0 to $+\infty$, and perform a shortest path algorithm in each level to obtain all Pareto optimal s-t-paths. However, since the problem is considered in continuous space, it is not clear how much we should increase λ to find a new extreme Pareto optimal path. The idea is to determine *events* which make changes in the shortest path during the continuously increasing λ process. Assume we perform a shortest path algorithm when $\lambda = 0$, and let π_1 be the resulted path with some breakpoints. Now we continuously increase λ until at least one of the breakpoints of π_1 changes in λ_c and a new path π_2 is generated. Regarding the definition, π_2 is an extreme Pareto optimal solution. Precisely, when $\lambda = \lambda_c$, the Pareto front changes and a jump to another Pareto front occurs. That is, to obtain a Pareto

optimal solution with clearance more than λ_c, it is needed that some brakpoints of π_2 are changed. Therefore, the number of such extreme Pareto optimal paths is equal to the number of these jumps.

3 Bi-objective PP Problem Among Polygonal Obstacles

In this section, we propose an algorithm to find the Pareto optimal solutions and analyze the complexity of this method. In the first subsection we describe the algorithm which is based on all the critical events that may change the shortest s-t-path. We discuss constructing data structures to handle these events. Finally, at the end of this section, we analyze the complexity of the algorithm.

3.1 The Algorithm

Given $\mathcal{P} = \{P_1, P_2, ..., P_m\}$ – m polygonal obstacles with total n vertices – as the input, we are supposed to find all extreme Pareto optimal paths. In order to find the shortest path for a specific clearance λ, a modified version of the Dijkstra's algorithm is used – called SP(λ,$\mathcal{V}$,$\mathcal{P}(\lambda)$). The inputs of this algorithm are the clearance value λ, the set of obstacles' vertices $\mathcal{V}$, and the set of fattened obstacles $\mathcal{P}(\lambda)$. Let T be the constructed tree by this algorithm. Whenever the SP algorithm is performed, T is initialized with the root s and the closest vertices to it as its children. Every node v in T is assigned a weight $w(v) > 0$ that shows the shortest distance between s and v, and has pointers to its parent and children. Obviously, the breakpoints of an s-t-path belong to the set of vertices $\mathcal{V}$ if $\lambda = 0$. It is clear when λ increases, the shortest s-t-path gets tangent to the fattened obstacles. For a polygonal obstacle P including a vertex v_i, let $P(\lambda)$ be the fattened obstacle and $TP_{v_i}(\lambda) = \{v_i^r(\lambda), v_i^l(\lambda), v_i^u(\lambda), v_i^d(\lambda)\}$ be the set of most four main directions, or say *tangent points*, of vertex v_i moving as a function based on λ (see Fig. 2 for an illustration). Since the shortest s-t-path is rectilinear, it can only be tangent to the $P_i(\lambda)$ at the tangent points of the vertices of it. For example, in Fig. 2, the path can be tangent to v_i at $v_i^l(\lambda_\epsilon)$, $v_i^u(\lambda_\epsilon)$, or both of them. An s-t-path touches at most three *tangent points* of a vertex (see Fig. 3).

Observation 1 *The tangent points of every vertex v_i can be computed as a linear function based on λ.*

Proof. Let $P(\lambda)$ be the fattened obstacle containing v_i as its vertex. Since this fattened obstacle is obtained by performing a Minkowski sum with the disc $\mathcal{D}(\lambda)$, the *tangent points* of v_i lie on the boundary of $\mathcal{D}(\lambda)$ in each of the four directions with the distance λ from v_i. Thus, the coordinates of the tangent points can be easily computed as follows:

$$\begin{cases} y_{v_i^r}(\lambda) = y_{v_i} \\ x_{v_i^r}(\lambda) = x_{v_i} + \lambda \end{cases}, \begin{cases} y_{v_i^l}(\lambda) = y_{v_i} \\ x_{v_i^l}(\lambda) = x_{v_i} - \lambda \end{cases}, \tag{1}$$

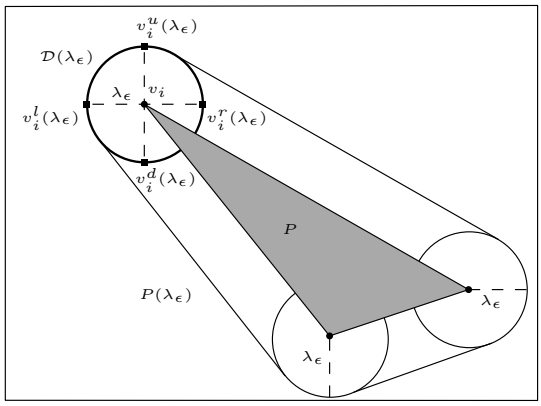


Fig. 2: Fattened obstacle $P(\lambda_\epsilon)$ and the *tangent points* of vertex v_i ($TP_{v_i}(\lambda_\epsilon)$).

$$\begin{cases} y_{v_i^u}(\lambda) = y_{v_i} + \lambda \\ x_{v_i^u}(\lambda) = x_{v_i} \end{cases}, \begin{cases} y_{v_i^d}(\lambda) = y_{v_i} - \lambda \\ x_{v_i^d}(\lambda) = x_{v_i} \end{cases} . \tag{2}$$

□

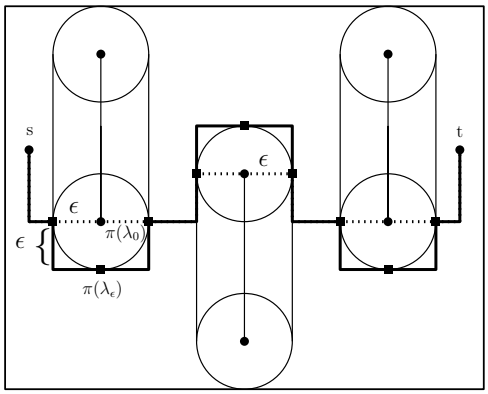


Fig. 3: The shortest s-t-path $\pi(\lambda_0 = 0)$ is shown in dotted chain and the shortest s-t-path $\pi(\lambda_\epsilon)$ is shown in solid bold chain.

The SP algorithm first creates $TP_{v_i}(\lambda)$ (for $0 \leqslant i \leqslant n$) according to equations (1) and (2). Next, it performs the Dijkstra's algorithm on the resulted tangent points to construct T. Note that the four mentioned tangent points of every vertex become equal to their origin when $\lambda = 0$. After performing the SP algorithm, all the tangent points are assigned a positive weight. Mitchell [10] proposed an appropriate planar subdivision method based on segment dragging techniques that can be used to find the adjacency relations between the nodes.

This part is adopted from [10]; however, to make the paper self-contained, we provide some basic notations here.

Let $S = (\theta, \Phi_1, \Phi_2)$ be a subdivision where a sweeping line is at inclination θ and it is being dragged parallel to itself whose endpoints lie on the rays l_1 and l_2 which have inclinations Φ_1 and Φ_2 respectively. By building the subdivisions $S = (3\pi/4, 0, \pi/2)$, $S = (5\pi/4, \pi/2, \pi)$, $S = (7\pi/4, \pi, 3\pi/2)$, and $S = (\pi/4, 3\pi/2, 2\pi)$, the closest points from any node under the L_1 metric are obtained. Therefore, for $O(n)$ vertices, the adjacent vertices can be found in $O(n \log n)$ time and $O(n)$ space.

Let π be the shortest path reported by the SP algorithm when $\lambda = 0$. As mentioned earlier, this path remains optimal until at least one of its breakpoints changes while λ increases. According to the definition, the new path $\pi^{'}$ is an extreme Pareto solution and it is accepted as a new optimal path. The following lemma shows that the change in the parent node of one of the breakpoint nodes can result in a new path.

Lemma 1. *During increasing λ, the optimal path does not change unless a change in the parents of some nodes in T occurs.*

Proof. Based on the *principle of optimality*, any sub-path of a shortest s-t-path is also a shortest path. So, let π be the current shortest path and $\pi^{'}$ be the new shortest path which is generated by the SP algorithm at some critical clearance λ_c. For the sake of contradiction, suppose that $\pi^{'}$ is obtained without any changes in T. Thus, if we backtrack $\pi^{'}$, the parents of the breakpoints remain unchanged. Apparently by moving backward from t to s in π, the parent node of each node is the closest node (based on the weights in the Dijkstra's algorithm) to it, and when no parent changes, this means the shortest path remains the same and this contradicts $\pi^{'}$ to be the new shortest s-t-path. □

According to this lemma, we have to keep track of changes in T (because it stores the parent and child relations and when a change in these relations occurs, T changes accordingly). Let $T(\lambda)$ be the tree at clearance λ. As mentioned previously, these changes are called *events*. We propose an algorithm to check all such events in $T(\lambda)$. We also construct data structures to handle them. The same as what is done in [2] and [11] every changes in T should be checked to find possible new paths. and it is obvious that not more than three types of events (where T changes) may occur while λ increases:

- *Type1:* A tangent point reaches a Manhattan weighted Voronoi diagram edge.
- *Type2:* Two tangent points reach the same x or y coordinate.
- *Type3:* Two obstacles intersect at some clearance λ_I.

In the following, we discuss these events. Let $MWVD(\mathcal{V})$ be the Manhattan weighted Voronoi diagram of V. This diagram partitions the plane into the regions each corresponding to a point p. Each point q that lies inside the region of p, is closer to point p compared to the other points. For the region corresponding to point p that contains all the points q. Thus, we have:

$$\forall p^{'} \in V, p^{'} \neq p \Rightarrow d(p, q) + w(p) \leq d(p^{'}, q) + w(p^{'}), \tag{3}$$

where $d(.,.)$ is the Manhattan distance between two points and $w(p)$ is the weight of point p. if p be the parent of p', we have:

$$w(p') = d(p, p') + w(p). \tag{4}$$

After performing the SP algorithm, a positive weight $w(v)$ is assigned to each node which represents the length of the shortest rectilinear path from s to v. According to the equation (4), the weight function of each node is a linear function that can be computed based on λ and the weight of its parent as an inline function. Note that tangent points move based on λ. The $MWVD(\mathcal{V})$ is constructed in $O(n \log n)$ time [8]. This diagram has some properties compared to the Euclidean Voronoi diagram. All the edges in the $MWVD(\mathcal{V})$ are horizontal, vertical or with a slope of $\pm\pi/4$. As λ increases, these edges move as a function of λ. For simplicity, edges between two points p and q and their equations as a function of λ are presented as follows.

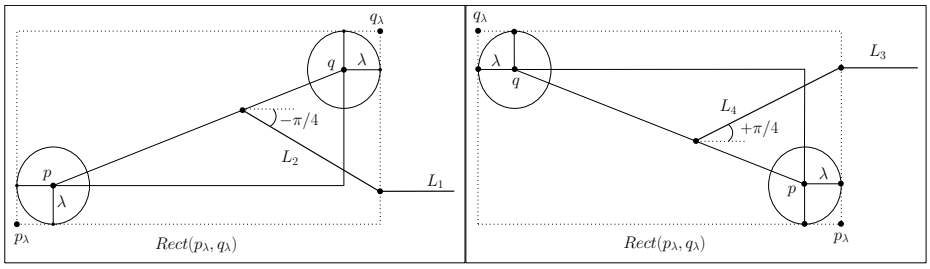
Let $Rect(p_\lambda, q_\lambda)$ be the rectangle cornered at $p_\lambda(x_p - \lambda, y_p - \lambda)$ and $q_\lambda(x_q + \lambda, y_q + \lambda)$ (see Fig. 4(a)) or $p_\lambda(x_p + \lambda, y_p - \lambda)$ and $q_\lambda(x_q - \lambda, y_q + \lambda)$ (see Fig. 4(b)). Fig. 4 shows two different positions of the two points p and q. For reasons of symmetry, we consider only one side and the lines on the other side can be computed easily with the same method. The equations are as follows:

$$\begin{cases} Y_{L_1} = -1/2x_q + 1/2((y_p + y_q) + x_p - (w(p) - w(q))), \\ X_{L_1} = x_q + \lambda, \end{cases} \tag{5}$$

$$Y_{L_2} = -X_{L_2} + 1/2(2\lambda + (x_p + x_q) + (y_p + y_q) - (w(p) - w(q))), \tag{6}$$

$$\begin{cases} Y_{L_3} = 1/2x_p + 1/2((y_p + y_q) - x_q - (w(p) - w(q))), \\ X_{L_3} = x_p + \lambda, \end{cases} \tag{7}$$

$$Y_{L_4} = X_{L_4} + 1/2(-2\lambda - (x_p + x_q) + (y_p + y_q) - (w(p) - w(q))). \tag{8}$$



(a) (b)

Fig. 4: Two different positions of the two points p and q and their $MWVD(\mathcal{V})$ edges.

Corollary 1. *The $MWVD(\mathcal{V})$ edges L_1 and L_2 (or L_3 and L_4) of two vertices $p(x_p, y_p)$ and $q(x_q, y_q)$ (see Fig. 4) intersect the $Rect(p_\lambda, q_\lambda)$ at the same point.*

Whenever a Type1 event occurs, an $MWVD(\mathcal{V})$ edge is passed. Of course, it is possible that a Type1 event occures and no parent changes. After constructing $MWVD(\mathcal{V})$, each vertex lies inside a cell of Voronoi diagram and the parent of each vertex is known. Critical λs are the intersection points of a vertex v with the edges of $MWVD(\mathcal{V})$ between the parent of v and its adjacent vertices. A Type1 event is shown in Fig. 5. The parent of the vertex p is pp before the event. When the intersection takes place, it means that the parent vertex should be changed to the new vertex qq in which p is going to enter its Voronoi region.

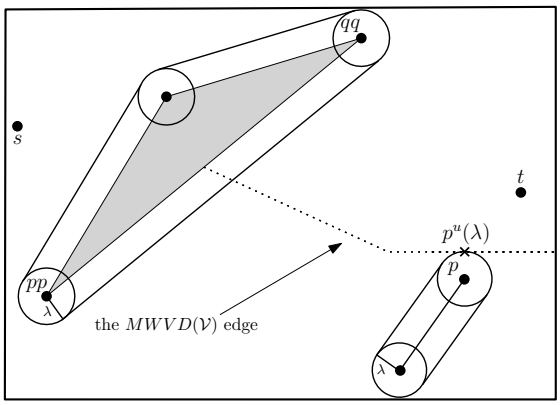


Fig. 5: A *Type*1 *event* between the fattened obstacles.

When two tangent points reach the same x or y coordinate, a Type2 event occurs. For simplicity, we introduce some definitions in order to find such critical λs. Let $H(v_i)$ and $V(v_i)$ be the horizontal and vertical lines passing through vertex v_i (for $0 \leqslant i \leqslant n$), respectively. Each $H(v_i)$ and $V(v_i)$ ends either at the first obstacle edge to which they are incident or at the vertex v_i (if it is not tangent to the obstacle that contains v_i). The lines with the latter characteristic, do not move when λ increases. Let $H_{V_i}(\lambda)$ and $V_{V_i}(\lambda)$ be the moving horizontal and vertical lines based on λ:

$$H_{V_i}(\lambda) = C_y + \lambda, \tag{9}$$

$$V_{V_i}(\lambda) = C_x + \lambda, \tag{10}$$

where C_y and C_x are the x and y coordinates of the points they pass through for $\lambda = 0$. Since this event occurs when two vertical or horizontal lines intersect, we can compute the critical λ at which they reach the same coordinate at a constant time. A Type2 event is shown in Fig. 6. The parent node of $p^u(\lambda)$ is $pp^u(\lambda)$ before two moving lines $H_q(\lambda)$ and $H_{pp}(\lambda)/H_p(\lambda)$ reach the same y coordinate. When the this event occurs, $q^b(\lambda)$ becomes the parent of $p^u(\lambda)$.

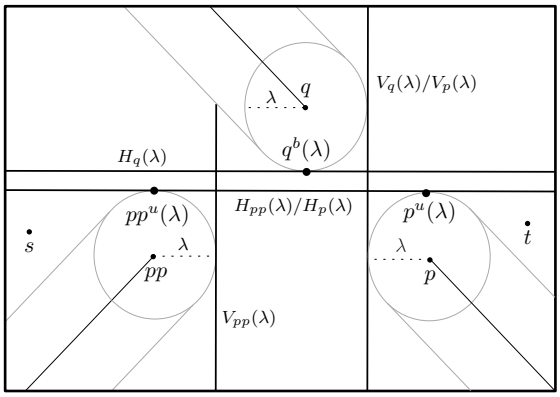


Fig. 6: A *Type*2 *event* between the fattened obstacles.

The Type3 event occur when two polygons intersect – called *intersection events*. For simplicity, assume that polygons do not intersect simultaneously and just two polygons intersect at each intersection event. Precisely, during increasing λ, when the polygons get fattened, they may intersect at some critical λs. When intersection events take place, the parents of some nodes can be changed (see Fig. 7 for an illustration). The critical λs at which intersection events occur, can be found with the Euclidean Voronoi diagram of $\mathcal{P}$. When two obstacles intersect, we unite them and consider them as one obstacle. Thus, each time an intersection occurs, the number of obstacles reduces by one. This type of event may change the parent of some nodes. A Type3 event is shown in Fig. 7. When $\lambda = 0$, the parent of v_4 is v_3 and the parent of v_5 is v_4. When the obstacles intersect, since the previous path is closed, the tangent point $v_5^b(\lambda)$ becomes the parent of $v_4^b(\lambda)$. The total complexity of the Euclidean Voronoi diagram of polygons, is $O(n)$ and it can be constructed in $O(n \log n)$ time [11, 4]. Since the Voronoi diagram of the polygons contains $O(n)$ edges [4], there are $O(n)$ critical λs at which polygons may intersect. Let $\mathcal{I} = \{I_{\lambda_1}, I_{\lambda_2}, ..., I_{\lambda_{O(n)}}\}$ be the set of the critical λs related to the intersection Type3 events.

Observation 2 *Since all the relations in T might be changed in the worst case when a Type3 event occurs, the problem of bi-objective path planning can be decomposed into $O(n)$ different clearance intervals. In each interval, the tree structure T is reconstructed, the weight function of each tangent point is computed, and the Type 1 and Type2 events can be handled.*

Any two consecutive members of the ordered set $\mathcal{I}$ construct an interval. More precisely, $[I_{\lambda_1}, I_{\lambda_2}), [I_{\lambda_2}, I_{\lambda_3}), ..., [I_{\lambda_{O(n)-1}}, I_{\lambda_{O(n)}}), [I_{\lambda_{O(n)}}, +\infty)$ are the set of such intervals. According to the observation 2 we consider each interval separately, and perform the following steps at the beginning of each interval until s and t are no more in the same connected component.

First, we perform the SP algorithm on the tangent points in order to construct T and obtain the weight of each node. As aforementioned, the weight of each

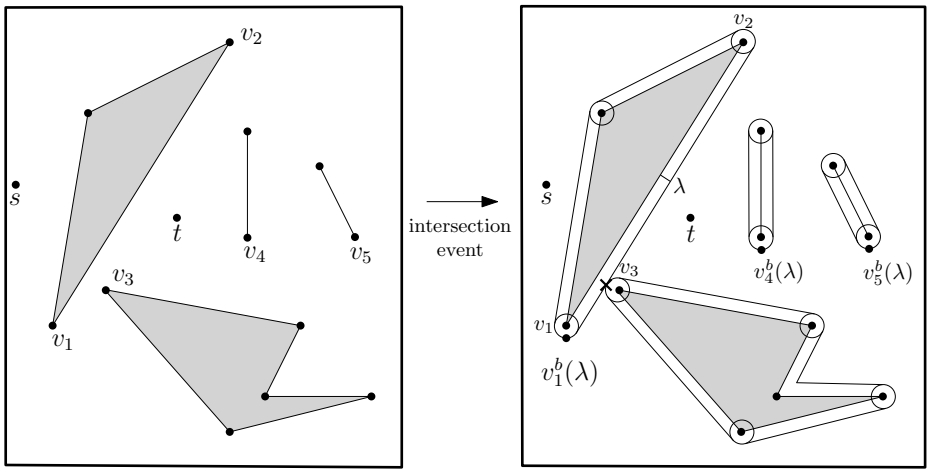


Fig. 7: The intersection of two obstacles that makes changes in T.

node is a linear function that is obtained based on λ and the weight of its parent. Note that once each node gets weighted, the $MWVD(\mathcal{V})$ is obtained based on equations (6)-(9) accordingly. It is obvious that the weight of the origin vertices can be obtained from the weight of tangent points according to equations (2) and (3) in order to compute the $MWVD(\mathcal{V})$.

Second, all the critical λs that cause a change in T related to Type1 and Type2 events are computed.

Lemma 2. *The intersection of a tangent point and an edge of $MWVD(\mathcal{V})$ (Type1 event) is efficiently computable.*

Proof. A Type1 event occurs when a tangent point intersects an edge of $MWVD(\mathcal{V})$. For simplicity, we consider horizonntal and vertical lines tangent to the obstacles at tangent points. When an obstacle intersects an edge of $MWVD(\mathcal{V})$, these lines and the edges intersect. Since all the moving lines – fixed lines and the edges of $MWVD(\mathcal{V})$ – are based on linear functions, the intersection points and such critical λ can be found in constant time for each event. □

Lemma 3. *The intersection of two horizontal or vertical lines associated to each vertex (Type2 event) is efficiently computable.*

Proof. As mentioned earlier, each vertex has two vertical and horizontal tangent lines. Depending on the obstacles and the location of the vertices, these lines can be moving or fixed. In both cases, the related equations are linear. Therefore, the critical λ, at which these lines reach the same x or y coordinate, can be found in constant time for each event. □

Third, we manage the data structures used to handle the events that make changes in T. Let H be the heap structure by which we store critical λs related

to Type1 events and Λ be the sorted list of critical λs associated with Type2 events. In each interval, the next critical λ, which should be handled, is the minimum λ that is extracted from H and Λ.

Assume p is a tangent point that intersects an $MWVD(\mathcal{V})$ edge between adjacent nodes pp and qq, and let pp be the parent of p. When the critical λ related to this intersection is extracted as minimum from H, we handle this Type1 event by changing the parent of p to qq. We delete all the λs related to the intersection of p and the other adjacent vertices of pp from H. Then, we insert critical λs related to the intersection of p and the adjacent vertices of qq to H. Since the parent node of p is changed, we update T by deleting the previous parent of p and inserting its new parent. We also update the initial weight of all nodes with p as their parent and the critical λs in H related to the $MWVD(\mathcal{V})$ edges associated with these nodes (e.g., the decrease key operation).

When a Type2 event occurs, i.e., the fixed or moving vertical or horizontal lines of two nodes p and q reach the same coordinate (w.l.o.g. for the horizontal lines assume p lies on the bottom of q and for the vertical lines assume p lies on the left of q). If the parent nodes of p and q are the same before intersection, we update the parent of q to p. Then, we update T by removing the previous parent of q and inserting p as its new parent. Changing the parent of q, changes its weight and subsequently the initial weight of all the nodes with q as their parent. Thus, the critical λs in H related to the edges of $MWVD(\mathcal{V})$ associated with these nodes should be updated sequentially.

3.2 Complexity Analysis

In this subsection we discuss the complexity of the proposed algorithm. In order to find the intersection events, we construct the Voronoi diagram of polygons under the Euclidean distance in $O(n \log n)$ time. Since $|\mathcal{I}| = O(n)$, the complexity of each interval is multiplied by $O(n)$.

Remark 1. The Dijkstra's algorithm runs in $O(n \log n + m)$ using a Fibonacci heap, where m is the number of edges of the graph. Since the graph is planar in our problem, m equals to $O(n)$ and the Dijkstra's algorithm runs in $O(n \log n)$ time.

Lemma 4. *The Type2 events can be handled in the worst case $O(n^3 \log n)$ time.*

Proof. Every vertex has a horizontal and a vertical tangent line. Thus, in the worst case, $O(n^2)$ events of this type can occur over all the intervals. When an event of such type occurs, it takes $O(n)$ time to update T and the weight of its children. We also update the edges of $MWVD(\mathcal{V})$ related to the nodes whose weights have been updated which cause updates in H. Let N_p be the number of neighbors of node p. The process of updating each edge of $MWVD(\mathcal{V})$ between the two nodes, takes a constant time. Updating the heap structure for each

vertex p also takes $O(N_p \log n)$. So,

$$A = \sum_{p \in \mathcal{V}} (N_p \log n) = \log n \sum_{p \in \mathcal{V}} N_p = O(n \log n).$$

Therefore, the total complexity time is $O(n^2 * n \log n)$, that is $O(n^3 \log n)$. □

Lemma 5. *The Type*1 *events can be handled in the worst case* $O(n^3 \log n)$ *time.*

Proof. According to the proposed algorithm, we insert the intersection point of a node to all of the $MWVD(\mathcal{V})$ edges corresponding to its parent adjacent nodes to H as critical λs. As the number of edges is $O(n)$, the total number of adjacent nodes is $O(n)$ as well. Thus, similar to lemma 4, it can be proved that the total complexity time for updating H is $O(n \log n)$ in each interval which is $O(n^2 \log n)$ in total. The same as lemma 4, we update T and the edges of $MWVD(\mathcal{V})$ that are between the nodes whose weights have been updated. By updating the edges of $MWVD(\mathcal{V})$, the heap structure H should be updated which can entirely be done in the worst case $O(n^2 \log n)$ for all events in each interval. Thus, the total complexity is $O(n^3 \log n)$. □

Therefore, we can conclude this section with the final result as follows.

Theorem 1. *The problem of bi-objective path planning among a set of polygonal obstacles with the objectives minimizing the length and maximizing the clearance, where the length and the clearance are measured using the Manhattan and the Euclidean metrics, respectively, can be solved in* $O(n^3 \log n)$*, where* n *is the total number of obstacles' vertices.*

4 An Approximation Algorithm for Euclidean bi-objective Path Planning

Consider the bi-objective PP problem with the objectives minimizing the length and maximizing the clearance under the Euclidean metric. We denote this problem with EbPP. Before going to a formal definition for an approximation for a bi-objective optimization problem, we denote that EbPP is a challenging open problem which follows some questions as:

1. How many extreme Pareto optimal paths are there in the worst case?
2. Does the problem of EbPP belong to the class of *Polynomial* problems?
3. What is the lower bound on the EbPP problem?

Based on our studies on this problem, when the length of the paths is measured using the Euclidean metric, more complicated algebraic weighted functions are replaced with the simple weighted functions presented in the previous section for the Manhattan case. So, we believe that the hardness of the problem of

EbPP is beyond the class of the polynomial problems. However, we guess the number of the extreme Pareto optimal solutions is $O(n^3)$ as well as the conjecture $\Omega(n^2)$ as the lower bound of the problem, where n is the number of total vertices of the polygonal obstacles. Therefore, as an initial approach, we present an approximation solution for the problem of EbPP.

The following definition for an (α, β)-*approximation* Pareto optimal solution is adopted from [2].

Definition 1. *Let Π be a bi-objective minimization problem with the objectives f_1 and f_2. A solution X is an (α, β)-approximation Pareto optimal solution for Π, if there is no solution Y such that $f_1(X) \geq \alpha f_1(Y)$ and $f_2(X) > \beta f_2(Y)$, or $f_1(X) > \alpha f_1(Y)$ and $f_2(X) \geq \beta f_2(Y)$.*

Remark 2. The proposed algorithm in the previous section which is optimal for the bi-objective path planning problem when the length of the path is measured using the Manhattan metric, provides $(\sqrt{2}, 1)$-approximation extreme Pareto optimal path for EbPP as well.

5 Conclusion

In this paper, we considered the problem of bi-objective path planning with the objectives minimizing the length and maximizing the clearance. We assumed a general case of the problem in the plane where the obstacles are a set of simple polygons with n vertices in total. We proposed an $O(n^3 \log n)$ time algorithm for finding all extreme Pareto optimal solutions of the problem where the length and clearance of the paths are measured using the Manhattan and Euclidean metrics, respectively. We also showed that such paths are good approximation solutions when both objectives are measured using the Euclidean metric. However, proposing an efficient algorithm to find the Pareto optimal paths for the latter problem remains a hard open problem as mentioned in Section four.

References

1. Clarkson, K., Kapoor, S., Vaidya, P.: Rectilinear shortest paths through polygonal obstacles in o (n (log n) 2) time. In: Proceedings of the third annual symposium on Computational geometry. pp. 251–257 (1987)
2. Davoodi, M.: Bi-objective path planning using deterministic algorithms. Robotics and Autonomous Systems **93**, 105–115 (2017)
3. Davoodi, M., Panahi, F., Mohades, A., Hashemi, S.N.: Clear and smooth path planning. Applied Soft Computing **32**, 568–579 (2015)
4. De Berg, M., Van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational geometry. In: Computational geometry, pp. 1–17. Springer (1997)
5. Geraerts, R.: Planning short paths with clearance using explicit corridors. In: 2010 IEEE International Conference on Robotics and Automation. pp. 1997–2004. IEEE (2010)
6. Ghosh, S.K., Mount, D.M.: An output-sensitive algorithm for computing visibility graphs. SIAM Journal on Computing **20**(5), 888–910 (1991)

7. Hershberger, J., Suri, S.: An optimal algorithm for euclidean shortest paths in the plane. SIAM Journal on Computing **28**(6), 2215–2256 (1999)
8. Hwang, F.K.: An o (n log n) algorithm for rectilinear minimal spanning trees. Journal of the ACM (JACM) **26**(2), 177–182 (1979)
9. Inkulu, R., Kapoor, S.: Planar rectilinear shortest path computation using corridors. Computational Geometry **42**(9), 873–884 (2009)
10. Mitchell, J.S.: L 1 shortest paths among polygonal obstacles in the plane. Algorithmica **8**(1-6), 55–88 (1992)
11. Wein, R., Van den Berg, J.P., Halperin, D.: The visibility–voronoi complex and its applications. Computational Geometry **36**(1), 66–87 (2007)