Shortest monotone descent path problem in polyhedral terrain ☆

Sasanka Roy, Sandip Das, Subhas C. Nandy *

Advanced Computing and Microelectronics Unit, Indian Statistical Institute, Kolkata 700 108, India

Received 2 December 2005; received in revised form 3 June 2006; accepted 24 June 2006

Available online 18 September 2006

Communicated by R. Klein

Abstract

Given a polyhedral terrain with n vertices, the shortest monotone descent path problem deals with finding the shortest path between a pair of points, called source (s) and destination (t) such that the path is constrained to lie on the surface of the terrain, and for every pair of points $p = (x(p), y(p), z(p))$ and $q = (x(q), y(q), z(q))$ on the path, if $dist(s, p) < dist(s, q)$ then $z(p) \geqslant z(q)$, where $dist(s, p)$ denotes the distance of p from s along the aforesaid path. This is posed as an open problem by Berg and Kreveld [M. de Berg, M. van Kreveld, Trekking in the Alps without freezing or getting tired, Algorithmica 18 (1997) 306–323]. We show that for some restricted classes of polyhedral terrain, the optimal path can be identified in polynomial time.

Keywords: Shortest path; Polyhedral terrain; Algorithm; Complexity

1. Introduction

The shortest path problem between two points s and t on the surface of an unweighted polyhedron is studied extensively in the literature. Sharir and Schorr [19] presented an $O(n^3 \log n)$ time algorithm for finding the geodesic shortest path between two points on the surface of a convex polyhedron with n vertices. Mitchell et al. [13] studied the generalized version of this problem where the restriction of convexity is removed. The time complexities of the proposed algorithms is $O(n^2 \log n)$. After a long time Chen and Han [6] improved the time complexity to $O(n^2)$. Finally, the best known algorithm for producing the optimal solution was proposed by Kapoor [8]; the running time of this algorithm is $O(n \log^2 n)$. Two approximation algorithms for this problem were proposed by Varadarajan and Agarwal [20]; it can produce paths of length $7(1 + \epsilon) \times opt$ and $15(1 + \epsilon) \times opt$ respectively; opt is the length of the optimal path between s and t, and ϵ is an user specified degree of precession. The running times are respectively $O(n^{5/3} \log(5n/3))$ and $O(n^{8/5} \log(8n/5))$. For convex polyhedron, an approximation algorithm was proposed by Agarwal et al. [1], which produces $(1 + \epsilon) \times opt$ solution, and runs in $O(n/\sqrt{\epsilon})$ time. A simple linear time 2-approximation algorithm for this problem is proposed by Hershberger and Suri [9].

The first work on approximating the minimum cost path of the weighted polyhedral surface appeared in a seminal paper by Mitchell and Papadimitrou [14]. It presents an $(1+\epsilon)$-approximation algorithm, that runs in $O(n^8 \log n)$ time. An implementable method for solving the minimum cost path problem was given by Mata and Mitchell [12], which formulates the problem as a graph search problem and assures a solution of length $(1+\epsilon) \times opt$. The running time of the algorithm is $O(\frac{n^3 N^2 W}{\epsilon w})$, where W and w are respectively the maximum and minimum weights among all the faces of the polyhedron. Efficient approximation algorithms are also available in [2,3,11,16,17]. The latest result on this problem appeared in the work of Aleksandrov et al. [4]. It proposes an $(1+\epsilon)$-approximation algorithm that runs in $O(C(P)\frac{n}{\sqrt{\epsilon}} \log \frac{n}{\epsilon} \log \frac{1}{\epsilon})$ time, where n and ϵ are as defined earlier, and $C(P)$ captures the geometric parameters and the weight of the faces of the polyhedron P.

Several variations of the path finding problems in polyhedral terrain are studied by Berg and Kreveld [5]. Given a polyhedral terrain $\mathcal{T}$ with n vertices, the proposed algorithm constructs a linear size data structure in $O(n \log n)$ time and can efficiently answer the following queries:

> Given a pair of points s and t on the surface of $\mathcal{T}$, and an altitude (height) ξ, does there exist a path between s and t such that for each point p on the path $z(p) \leqslant \xi$?
>
> Given a pair of points s and t on $\mathcal{T}$, determine the minimum total ascent/descent among the path(s) between s and t, where the *total ascent* of a non-monotone path is defined in [5].

Recently an interesting variation of the shortest path problem in the context of a terrain is proposed by Mitchell and Sharir [15], where the objective is to compute the L_1-shortest path between a pair of given points, such that the path is restricted to lie on or above a given polyhedral terrain with n faces. The proposed algorithm runs in $O(n^3 \log n)$ time. The same paper also studies another variation of the shortest path problem on a terrain like structure, where a set of n vertical walls parallel to the x-axis are given. Each wall is positioned on the xy-plane. The ith wall, is positioned at $y = a_i$, and its top boundary (denoted by e_i) is a line of the form $z = b_i x + c_i$, where a_i, b_i and c_i are given constants, $a_1 < a_2 < \cdots < a_n$. The objective is to report the L_2-shortest path between a given pair of query points s and t, where $s < a_1$ and $t > a_n$. The problem is referred to as the *L_2-shortest path over walls*. Note that the shortest path is always monotone with respect to the y-axis, and it bends on the edges e_i, $i = 1, 2, \ldots, n$. It is also proved that the shortest path from s to t is the concatenation of two sub-paths, one of them is monotone ascending and the second one is monotone descending with respect to z-coordinate. The standard method of solving this problem involves a preprocessing phase which splits each edge e_i into segments, and then defines the shortest path map [18], such that the optimal L_2-path from s to t can be obtained by following an appropriate path in the map. In [15], it is proved that the size of the shortest path map is $O(n^2)$ in the worst case, but finding a polynomial time algorithm for constructing the map is left as an open problem.

We address the problem of computing the shortest among all possible monotone descending paths (if at least one such path exists) between a pair of points on the surface of a polyhedral terrain. This is a long-standing open problem in the sense that no bound on the combinatorial or Euclidean length of the shortest monotone descent path between a pair of points on the surface of a polyhedral terrain is available in the literature [5]. Some interesting observations of the problem lead us to design efficient polynomial time algorithm for solving this problem in the following two special cases, where

1. our search domain is among all possible monotone descent paths from s to t which are constrained to pass through a sequence of faces such that each pair of consecutive faces are in convex position, and the objective is to identify the shortest among such paths (see Section 4), and
2. given a sequence of pairwise adjacent faces having their boundaries parallel to each other (but the faces are not all necessarily in convex (respectively concave) position); the objective is to find the shortest monotone descent path from s to t through that sequence of faces (see Section 5).

In Case 1, if the terrain contains n triangulated faces, the preprocessing of those faces need $O(n^2 \log n)$ time and $O(n^2)$ space, and the shortest monotone descent path query through a sequence of convex faces can be answered in $O(k + \log n)$ time (provided such a path exists), where k is the number of faces through which the optimum path passes. In Case 2, if a sequence of n faces with their boundaries in parallel position is given, the shortest monotone

descent path from s to t through that face sequence can be computed in $O(n \log n)$ time. The solution technique for this case indicates the hardness of handling the general terrain.

The problem is motivated from the agricultural applications where the objective is to lay a canal of minimum length from the source of water at the top of the mountain to the ground for irrigation purpose. Another application of Case 2 can be observed in the design of fluid circulation systems in automobiles or refrigerator/air-condition machines.

2. Preliminaries

A terrain $\mathcal{T}$ is a polyhedral surface in $\mathbb{R}^3$ with a special property: the vertical line at any point on the xy-plane intersects the surface of $\mathcal{T}$ at most once. Thus, the projections of all the faces of a terrain on the xy-plane are mutually non-intersecting at their interior. Each vertex p of the terrain is specified by a triple $(x(p), y(p), z(p))$. Without loss of generality, we assume that all the faces of the terrain are triangles, and the source point s is a vertex of the terrain.

Definition 1. [13] Let f and f' be a pair of faces of $\mathcal{T}$ sharing an edge e. The *planar unfolding* of face f' onto face f is the image of the points of f' when rotated about the line e onto the plane of f such that the points in f and the points in f' lie in two different sides of the edge e respectively (i.e., the faces f' and f become coplanar and they do not overlap after unfolding).

Let $\{f_0, f_1, \ldots, f_m\}$ be a sequence of adjacent faces. The edge common to f_{i-1} and f_i is e_i. We define the *planar unfolding with respect to the edge sequence* $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$ as follows: obtain the planar unfolding of face f_m onto face f_{m-1}, then get the planar unfolding of the resulting plane onto f_{m-2}, and so on; finally, get the planar unfolding of the entire resulting plane onto f_0. From now onwards, this event will be referred to as $U(\mathcal{E})$.

A path $\pi(s,t)$ from a point s to a point t on the surface of the terrain is said to be a geodesic path if it entirely lies on the surface of the terrain, it is not self-intersecting, and in each face its intersection with the path $\pi(s,t)$ is a straight line segment. The geodesic distance $dist(p, q)$ between a pair of points p and q on $\pi(s,t)$ is the length of the path from p to q along $\pi(s,t)$. The path $\pi_{\text{geo}}(s,t)$ is said to be the *geodesic shortest path* if the distance between s and t along $\pi_{\text{geo}}(s,t)$ is minimum among all possible geodesic paths from s to t.

Lemma 1. [13] *For a pair of points α and β, if $\pi_{\text{geo}}(\alpha, \beta)$ passes through an edge sequence $\mathcal{E}$ of a polyhedron, then in the planar unfolding $U(\mathcal{E})$, the path $\pi_{\text{geo}}(\alpha, \beta)$ is a straight line segment.*

Definition 2. A path $\pi(s,t)$ $(z(s) \geqslant z(t))$ on the surface of a terrain is a *monotone descent path* if for every pair of points $p, q \in \pi(s,t)$, $dist(s, p) < dist(s, q)$ implies $z(p) \geqslant z(q)$.

We will use $\pi_{\text{md}}(p,q)$ and $\delta(p,q)$ to denote the shortest monotone descent path from p to q and its length, respectively. If $p*$ and $q*$ are the image of p and q respectively in the unfolded plane along an edge sequence traversed by path $\pi_{\text{geo}}(p,q)$, the path $\pi_{\text{geo}}(p,q)$ corresponds to the line segment $[p^*, q^*]$ in that unfolded plane, and it satisfies monotone descent property, then q is said to be *straight line reachable* from p in the unfolded plane. In such a case, $\pi_{\text{md}}(p,q) = \pi_{\text{geo}}(p,q)$.

Remark 1. A monotone descent path between a pair of points s and t may not exist (Fig. 1(a)). Again, if monotone descent path from s to t exists, then $\pi_{\text{md}}(s,t)$ may not coincide with $\pi_{\text{geo}}(s,t)$ (Fig. 1(b)).

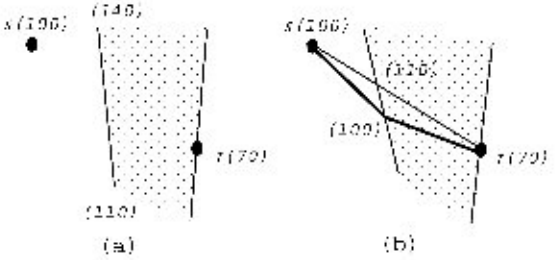

Fig. 1. Justification of Remark 1.

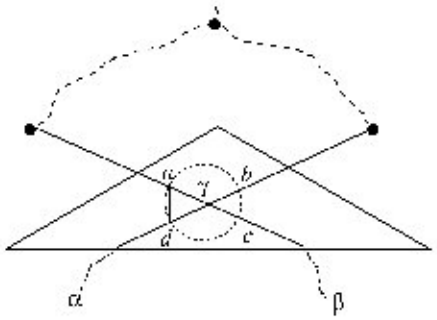

Fig. 2. Proof of Lemma 3.

Lemma 2. *If the shortest monotone decent path $\pi_{md}(s,t)$ passes through a face f, then the intersection of $\pi_{md}(s,t)$ with the face f is line segment.*

Proof. [By contradiction] Let the portion of $\pi_{md}(s,t)$, which lies in face f, is not a single line segment. Let us consider a pair of points $p_1, p_2 (\in f)$ on the path $\pi_{md}(s,t)$ (with $dist(s,p_1) < dist(s,p_2)$) such that their joining line segment does not coincide with an edge of $\pi_{md}(s,t)$. Note that the line segment $[p_1, p_2]$ satisfies the monotone descent property, and its length is less than the length of the path from p_1 to p_2 along $\pi_{md}(s,t)$. Hence we have a contradiction. □

Lemma 3. *Given a vertex s and a pair of points α and β on the terrain $\mathcal{T}$, $\pi_{md}(s,\alpha)$ and $\pi_{md}(s,\beta)$ cannot intersect except at some vertex of $\mathcal{T}$. Moreover, if they intersect at a vertex v then the length of the subpath from s to v on both $\pi_{md}(s,\alpha)$ and $\pi_{md}(s,\beta)$ are same.*

Proof. Let $\pi_{md}(s,\alpha)$ and $\pi_{md}(s,\beta)$ intersect at a point γ, which is equidistant from s along both the paths $\pi_{md}(s,\alpha)$ and $\pi_{md}(s,\beta)$, otherwise one of these two paths cannot be optimum. Thus, the second part of the lemma follows. We prove the first part by contradiction. We need to consider two cases: (i) γ is inside a face (say f), and (ii) γ lies on an edge e which is adjacent to a pair of faces f and f'.

In case (i) consider a very small circle centered at γ which completely lies inside the face f. The path $\pi_{md}(s,\alpha)$ intersects the circle at two points b and d, and the path $\pi_{md}(s,\beta)$ intersects the circle at a and c (see Fig. 2). From the second part of the lemma, the length of the paths $s \sim b \to \gamma \to d \sim \alpha$ and $s \sim a \to \gamma \to d \sim \alpha$ are same, and both are optimum paths (by the statement of the lemma). Now, consider the path $s \sim a \to d \sim \alpha$. Its length is less than both the paths mentioned above (by triangle inequality). Again, $z(a) \geqslant z(\gamma) \geqslant z(d)$ due to the fact that both $\pi_{md}(s,\alpha)$ and $\pi_{md}(s,\beta)$ are monotone descent. So, the path $s \sim a \to d \sim \alpha$ is monotone descent also. Thus, we have a contradiction.

The case (ii) can be similarly handles by unfolding f onto f' and drawing the circle around γ in the unfolded plane. □

Definition 3. Given an arbitrary point p on the surface of the terrain $\mathcal{T}$, the descent flow region of p (called $DFR(p)$) is the region on the surface of $\mathcal{T}$ such that each point $q \in DFR(p)$ is reachable from p through a monotone descent path.

Given a polyhedral terrain $\mathcal{T}$ and a given point $s \in \mathcal{T}$, we study the following problems:

P1: Construct $DFR(s)$.
P2: For a given query point $t \in DFR(s)$ report $\pi_{md}(s,t)$, and its length.

Problem P2 seems to be difficult in general. We identified the following two special cases where it can be solved in polynomial time.

P2.1: For a given source point s, we can construct a data structure such that given any query point $t \in DFR(s)$, we can identify the shortest monotone descent path from s to t provided $DFR(s)$ is convex (to be defined in Section 4).

P2.2: Given a sequence of faces $\{f_0, f_1, \ldots, f_m\}$ of a polyhedral terrain (not necessarily convex/concave), if e_i denotes the edge separating f_{i-1} and f_i, and the projections of the edges $e_1, e_2, \ldots, e_m$ on the XY-plane are parallel, then we can identify the monotone descent shortest path between a pair of points $s \in f_0$ and $t \in f_m$ through that sequence of faces.

3. Computation of $DFR(s)$

Given the source point s, if it lies on an edge e of a triangulated face, we add s with the vertex opposite to e in both the faces adjacent to e. If s lies inside a triangulated face then we add s to all the three vertices of that face. Thus s may always be considered as a vertex of the triangulated terrain.

Observation 1. If r is reachable from s using monotone descent path, then $DFR(r) \subseteq DFR(s)$.

Observation 2. Let Δspq be a triangular face adjacent to the source s with $z(p) \leqslant z(q)$. Now,

(i) if $z(s) \geqslant z(q)$ then $\Delta spq \subseteq DFR(s)$,
(ii) if $z(s) < z(p)$ then $\Delta spq \cap DFR(s) = \phi$ (empty region), and
(iii) if $z(p) \leqslant z(s) < z(q)$ then there exists a point r on the edge (p, q) (with $z(r) = z(s)$) such that $\Delta spr \subseteq DFR(s)$, and $\Delta srq \not\subset DFR(s)$. In this case, if $z(p) = z(r)$, then Δspr degenerates to the line segment $[s, p]$ (or equivalently $[s, r]$).

In Fig. 3, faces B and F satisfy Cases (i) and (ii) of Observation 2 respectively; all other faces satisfy Case (iii) of Observation 2.

We consider all the faces adjacent to s, and compute the initial descent flow region inside those faces. The union of these regions is denoted as $IDFR(s)$. The projection of $IDFR(s)$ on the XY-plane is a connected region, which may be (i) a simple polygon with s inside its kernel, or (ii) a collection of convex polygons each having s as a vertex. The vertices of $IDFR(s)$ (excluding s itself) are said to be the descent flow neighbors of s, and are denoted as $DFN(s)$ (see Fig. 3).

Lemma 4. $DFR(s) = (\bigcup_{r \in DFN(s)} DFR(r)) \cup IDFR(s)$.

Proof. From Observations 1 and 2, $(\bigcup_{r \in DFN(s)} DFR(r)) \cup IDFR(s)) \subseteq DFR(s)$. We now prove that $DFR(s) \subseteq (\bigcup_{r \in DFN(s)} DFR(r)) \cup IDFR(s)$.

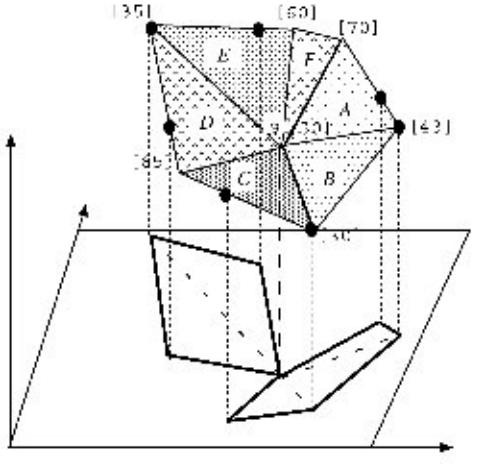

Fig. 3. Illustration of $DFN(s)$.

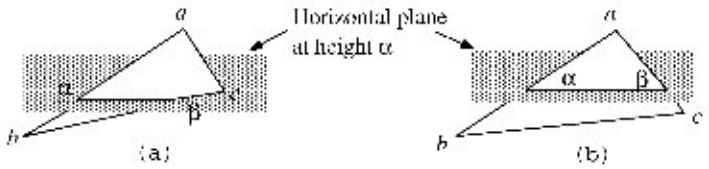

Fig. 4. *DFR* of a point α on an edge (a, b) where (a) $z(c) > z(\alpha)$ and (b) $z(c) < z(\alpha)$.

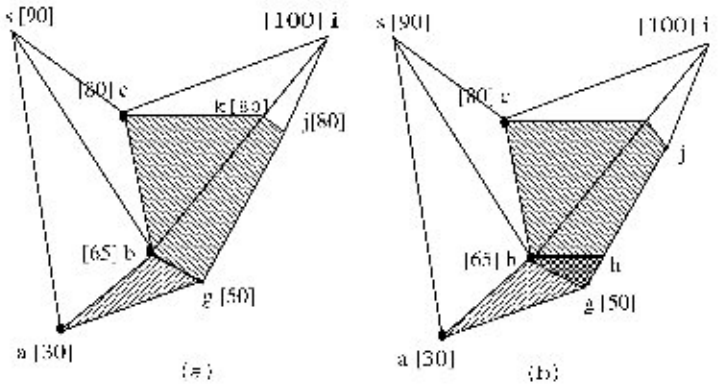

Fig. 5. Order of processing of the *DFN*s'.

Let q be a point in $DFR(s)$ but not in $(\bigcup_{r \in DFN(s)} DFR(r)) \cup IDFR(s)$. Consider a monotone descent path from s to q. By Observation 2, it intersects a boundary edge $[a, b]$ of $IDFR(s)$ at a point c. Assume that $z(a) \geqslant z(b)$. The path from the point a to the point c along the boundary $[a, b]$ is a monotone descent path. Thus, $q \in DFR(a)$, which leads to the contradiction. □

We compute $IDFR(s)$ by considering the faces adjacent to s. The processing of the triangular faces which are not adjacent with the source s is discussed below.

Observation 3. If more than one point on the boundary of a face Δabc are reachable from s, then for each pair of such points α and β, $z(\alpha) < z(\beta)$ implies $DFR(\alpha) \cap \Delta abc \subseteq DFR(\beta) \cap \Delta abc$, and vice versa.

Observation 4. The intersection of a face of $\mathcal{T}$ with $DFR(s)$ may be a vertex of that face, an edge of that face which is parallel to XY-plane, or a polygonal region.

During the execution of the algorithm, we maintain a priority queue $\mathcal{Q}$ which is initialized with $DFN(s)$, and process these elements in an ordered manner (as discussed below). During the processing of a member $\alpha \in \mathcal{Q}$, the set of points $DFN(\alpha)$ are also inserted in $\mathcal{Q}$. The algorithm continues until all the members in $\mathcal{Q}$ are processed.

While processing an element $\alpha \in \mathcal{Q}$, if it is a vertex of a triangular face Δabc, then it is processed as in Lemma 4. If it appears in the middle of an edge (a, b) (assuming $z(a) > z(b)$) then two situations may arise.

- If $z(c) > z(\alpha)$ then there exists a point β on the edge (b, c) with $z(\beta) = z(\alpha)$. Here, the triangular region $\Delta b\alpha\beta$ is included in $DFR(s)$ (see Fig. 4(a)).
- If $z(c) < z(\alpha)$ then there exists a point β on the edge (a, c) with $z(\beta) = z(\alpha)$. Here the quadrilateral $\square b\alpha\beta c$ is included in $DFR(s)$ (see Fig. 4(b)).

In order to explain the order of processing of the elements in $\mathcal{Q}$, let us consider a terrain in Fig. 5, the z-coordinates of all the vertices are given in square bracket, and the $DFN(s)$'s are marked with dark circles. Now, consider the following situations:

c is processed prior to b: Here, after processing of c, the region Δcbk is included in $DFR(s)$, and generates the point k as a new DFN. After processing k, $\square kbgj$ is included in $DFR(s)$. Next, when b is processed, Δbga is included in $DFR(s)$ (see Fig. 5(a)).

b is processed prior to c: Here, after processing of b, Δabg, and Δbgh ($\subset \Delta bgi$) are included in $DFR(s)$. Now, if we process the point c then, as mentioned above, the $\square cbgj$ is included in $DFR(s)$ (see Fig. 5(b)).

In the latter situation, Δbgh will be included twice in $DFR(s)$. This situation can be avoided by (i) processing the DFNs' in decreasing order of their z-coordinates using a priority queue, and (ii) maintaining a *flag* with each face which will be set to the value "1" if it is considered during the processing of a DFN. Observation 3 along with the above discussion lead to the following algorithm which identifies $DFR(s)$ for a given source vertex s, and stores it in the form of a doubly connected edge list. We also maintain a data structure $PSLG$ for planar point location query [10] using the projections of the faces in $DFR(s)$ on the XY-plane.

Algorithm.
Input: A triangulated polyhedral terrain, and the source s.
Output: $DFR(s)$ in the form of doubly connected edge list $DCEL$, and a planar point location data structure $PSLG$.
Data structure: A priority queue Q to store the unprocessed DFNs' in decreasing order of their z-coordinates.

```
begin
 put s in Q;
 while Q is not empty do
   p = Q(1);
   for each face f attached to p do
    if flag of face f is not equal to 1 then
      compute DFN(p) in face f and insert them in Q;
      set flag of face f to 1;
    endif
   endfor
   compute the faces in DFR(p) attached with point p;
   insert each of the faces in the data structure DCEL
 endwhile
 Use DCEL to construct the PSLG data structure [10]
end.
```

Lemma 5. *The proposed algorithm processes each face at most once, and outputs $DFR(s)$ correctly.*

Proof. The first part of the lemma follows from the use of flag bit during the processing.

For the second part, consider a portion of a face f in the descent flow region of s but is not included in $DFR(s)$ by our algorithm. Let p be a point in this region having maximum z-coordinate. Surely, p is on an edge of f, and p is not processed as a DFN from Q. Note that the flow can reach from s to p through a face f' (which is adjacent to f), which is also not included in $DFR(s)$. We can apply the same argument repeatedly to prove that s is not inserted in Q. Thus, we have a contradiction. □

Theorem 1. *The proposed algorithm for computing $DFR(s)$ needs $O(n \log n)$ time and $O(n)$ space, and given an arbitrary point t, it searches in the $DFR(s)$ data structure in $O(\log n)$ time to report whether a monotone descent path exists from s to t along the surface of $\mathcal{T}$.*

Proof. Each face is processed at most once for inclusion in $DFR(s)$ (by Lemma 5), and while processing each (triangulated) face at most three DFNs' are generated (see Fig. 5). Thus, total number of DFNs' inserted in Q is $O(n)$ in the worst case. Inserting a part of a face in $DCEL$ requires $O(1)$ time. Since a single operation in a priority queue needs $O(\log n)$ time, $DCEL$ can be constructed in $O(n \log n)$ time. The same argument leads to the fact that $DCEL$

needs $O(n)$ space. Given the *DCEL*, the *PSLG* data structure can also be constructed in $O(n \log n)$ time using $O(n)$ space [10]. The query time complexity in *PSLG* is $O(\log n)$ [10]. □

4. Shortest monotone descent path in convex *DFR*

We now study a restricted version of the descent flow problem where $DFR(s)$ is convex.

Definition 4. Let f and f' be two adjacent faces of the terrain $\mathcal{T}$ sharing an edge e. Let p and q be two points on faces f and f' respectively (none of p and q is on e). Now, if the line segment $[p, q]$ is not visible (respectively visible) from the outside of the terrain then f and f' are said to be in convex (respectively concave) position.

Definition 5. Given a terrain $\mathcal{T}$ and a source point s, $DFR(s)$ is said to be *convex* if every two adjacent faces in $DFR(s)$ is in convex position. Similarly, $DFR(s)$ is said to be *concave* if its every pair of adjacent faces is in concave position.

We now study the properties of shortest monotone descent path in a convex $DFR(s)$. The convexity of $DFR(s)$ can be tested very easily by observing the neighbors of its each face. From now onwards, we assume that the $DFR(s)$ on which we are working, is convex.

Observation 5. If p_1, p_2 are two points on a face of $\mathcal{T}$, and p_3 is another point on the line segment $[p_1, p_2]$, then $z(p_1) > z(p_3)$ implies $z(p_2) < z(p_3)$.

Lemma 6. *Let f and f' be two adjacent faces of a polyhedral terrain which are in convex position. The edge $e = (a, b)$ separates f and f'. Consider a pair of points p and q on faces f and f' respectively, and a point c on e with $z(p) = z(c)$.*

(a) *Now the edge e can be partitioned into two parts $[a, c]$ and $(c, b]$ such that the descent flow from p to the face f' is possible through the portion $[a, c] \in e$ but not possible through the portion $(c, b]$.*
(b) *Let q^* denote the image of the point q in the planar unfolding of f' onto f. Now,* (i) *if the line segment $[p, q^*]$ intersects the line segment $[a, c]$ $(\in e)$ in the unfolded plane, then $q \in DFR(p)$ and the geodesic shortest path from p to q through the edge e is the shortest monotone decent path from p to q, and* (ii) *if $[p, q^*]$ intersects the line segment $(c, b]$ but $q \in DFR(p)$, then $[p, c] + [c, q]$ forms the shortest monotone descent path from p to q through the edge e.*

Proof. Part (a) of the lemma is trivial. We now prove part (b) of the lemma.

Let $\pi_{geo}(p, q; e)$ denote the geodesic shortest path from p to q passing through the edge e. If the line segment $[p, q^*]$ (in the unfolded plane) intersects e (at a point, say η) in its interior, then by Lemma 1, the image of $\pi_{geo}(p, q; e)$ in the unfolded plane coincides with the line segment $[p, q^*]$. Now, two cases need to be considered: (1) $z(\eta) \leqslant z(p)$ and (2) $z(\eta) > z(p)$.

Case 1. Here, by Observation 5, $z(q^*) < z(\eta)$. As the two faces f and f' are in convex position, $z(q) \leqslant z(q^*)$. Thus both the line segments $[p, \eta]$ and $[\eta, q]$ are monotone descent (see Fig. 6(a)), and part (i) of the lemma follows.

Case 2. Here the line segment $[p, \eta]$ is not monotone descent in the plane f. Consider any monotone descent path from p to q which intersects the line segment $[a, c]$ (at a point, say η'). Note that the length of such a path remains same as that of its image in the unfolded plane, and it attains minimum when $\eta' = c$ as illustrated in Fig. 6(b). This proves part (ii) of the lemma. □

Let v be a vertex of $\mathcal{T}$ and p be a point in $DFR(v)$ which is reachable from v through a sequence of edges $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$ of $DFR(v)$; the faces f_{i-1} and f_i, attached to edge e_i, are in convex position; $v \in f_0$, $p \in f_m$. Now, let R^* denote the region obtained by the planar unfolding $U(\mathcal{E})$, which is a polygonal region in the unfolded plane. Now we have the following result:

Lemma 7. *If p^* denotes the image of the point p in R^*, and the line segment $[v, p^*]$ completely lies inside R^*, then the path $\pi(v, p)$ on $\mathcal{T}$, whose image in R^* is the line segment $[v, p^*]$, is the shortest monotone descent path from v to p through the faces $\{f_0, f_1, f_2, \ldots, f_m\}$.*

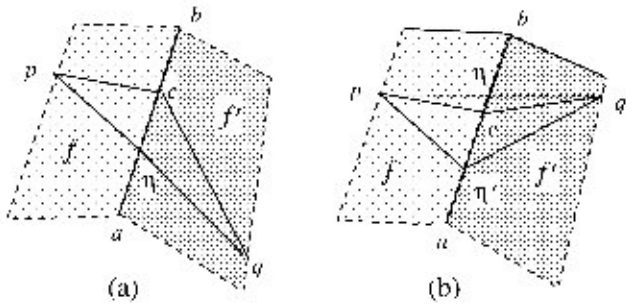

Fig. 6. Proof of Lemma 6.

Proof. From Lemma 1, the path $\pi(v, p)$ is a shortest geodesic path through $\{f_0, f_1, f_2, \ldots, f_m\}$. Next, we show that $\pi(v, p)$ is a monotone descent path.

Let c_i be the point of intersection of $\pi(v, p)$ with the edge e_i. Since $\pi(v, p)$ passes through $DFR(v)$, $z(c_i) < z(v)$ for all $i = 1, \ldots, m$. Rename $v = c_0$ and $p = c_{m+1}$. Now, by repeated application of the proof technique of Lemma 6, it can be shown that $z(c_0) > z(c_1) > z(c_2) > \cdots > z(c_m) > z(c_{m+1})$. □

Remark 2. If the shortest monotone descent path from a vertex v to a point p is obtained as in Lemma 7, then the point p is *straight-line reachable* from the vertex v.

Remark 3. Let e be an edge of $\mathcal{T}$ separating the faces f and f'. A point p on e may be straight line reachable from a vertex v through different edge sequences. Thus, p may be straight line reachable from v through both f and f'.

Definition 6. A point α on a line segment $[a, b]$ (portion of an edge) is said to be the *frontier point* with respect to a vertex v if α is straight line reachable from v through an edge sequence $\mathcal{E}$ and it is the closest point to v on the line segment $[a, b]$.

It is easy to see that α can be either a or b or the perpendicular projection of v on the line segment $[a, b]$ in the planar unfolding R^*.

The above discussions lead to a preprocessing step of $DFR(s)$ similar to [13]. It splits each face f of $DFR(s)$ into *homogeneous partitions* such that for every point p in a partition the shortest monotone descent path from s reaches p through the same edge sequence. Note that each of these partitions must be maximal in the sense that it is not properly contained in some other *homogeneous partition* of f. We will refer the data structure storing this homogeneous partitions of $DFR(s)$ by *HDFR*.

Definition 7. A segment $I = [a, b]$ on an edge $e \in DFR(s)$ is said to be a *homogeneous segment* (or *h-segment* in short) if for every point $\alpha \in I$, the shortest monotone descent path from s to α passes through the same edge sequence.

4.1. Preprocessing

Our algorithm for finding shortest monotone descent path on convex $DFR(s)$ creates a data structure *HDFR* in two phases. In Phase 1, each edge $e = (a, b)$ of $DFR(s)$ is split into *h-segments* $\{I_i = [a_i, a_{i+1}],\ i = 0, \ldots, k-1\}$, $a_0 = a$, $a_k = b$, $\bigcup_{i=0}^{k-1} I_i = e$. The points $a_0, a_1, a_2, \ldots, a_k$ are referred to as *break-points*. In Phase 2, the interior of each face of $DFR(s)$ is split into homogeneous partitions (similar to Voronoi partition). Below, we describe Phase 1 and Phase 2 in detail. The *HDFR* data structure is similar to the data structure for storing $DFR(s)$ as defined in Section 3; but its each edge e in the *DCEL* data structure is attached with an associated structure $AVL(e)$ which is defined in Phase 1, and its each face $f = \Delta abc$ in the *DCEL* data structure is attached with the homogeneous partition $VOR(f)$ inside Δabc, which is basically the Voronoi diagram of a set of weighted points, and is explained in Phase 2.

Phase 1. Let p be a point on the surface of $\mathcal{T}$ which is straight-line reachable from a vertex $r \in DFR(s)$. The shortest monotone descent path from s to p passing through the vertex r, denoted by $\pi_r(p)$, is the concatenation of $\pi_{\mathrm{md}}(s, r)$

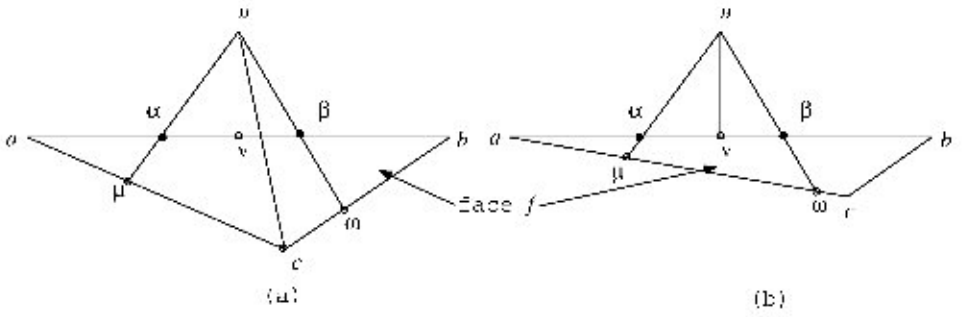

Fig. 7. Processing a point v which is not a vertex of $DFR(s)$.

and the line segment $[r, p]$. Its length is $\delta_r(p) = \delta(s, r) + dist(r, p)$. Here $dist(r, p)$ is equal to the length of the straight line segment $[r, p^*]$ in the unfolded plane.

Definition 8. Let $I = [a, b]$ be an *h-segment* on an edge e such that $\pi_{md}(s, \alpha) = \pi_r(\alpha)$ for every point $\alpha \in I$, then the vertex r is said to be the *link-vertex* for the *h-segment* I.

The end points of the *h-segments* are referred to as *break-points*. If r is the *link-vertex* of a *h-segment* I, and $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$ be the edge sequence which are intersected by the line segment $[r, \alpha]$ for every point $\alpha \in I$, then the last edge e_m in $\mathcal{E}$ is called the predecessor of I in the *HDFR* data structure. If $[r, \alpha]$ does not intersect any edge, then the predecessor of I is r itself. Then we have the following remarks.

If a_i is a break-point on an edge e, and is shared by two *h-segments* $[a_{i-1}, a_i]$ and $[a_i, a_{i+1}]$ with link vertices r and u respectively, then $\delta_r(a_i) = \delta_u(a_i)$.

If $I_1 = [a_1, b_1]$ and $I_2 = [a_2, b_2]$ are two *h-segments* on an edge e (adjacent to faces f and f') with *link-vertex* r_1 and r_2 respectively, and both I_1 and I_2 are reachable from r_1 and r_2 respectively through the same face f, then I_1 and I_2 have mutually disjoint interiors.

Thus, the *h-segments* generated on an edge e are non-overlapping, and can be orderly maintained in an *AVL*-tree, named as $AVL(e)$. We also need to compute the *frontier-point* on each *h-segment* $[a, b]$ with respect to its *link-vertex*. Each *h-segment* is attached with its (i) *predecessor*, (ii) *link-vertex*, and (iii) three pointers, namely ptr_1, ptr_2 and ptr_3, which will point to three elements of MIN-HEAP data structure corresponding to the two *break-points* and the *frontier point* of that *h-segment*. A vertex of $DFR(s)$ in the *HDFR* data structure is also attached with its *predecessor* and *link-vertex*, which can be defined in a manner similar to the *h-segments*.

During the execution of the *HDFR* creation algorithm, we use a MIN_HEAP containing all the vertices, *break-points* and *frontier-points* explored so far. Each element α in the MIN_HEAP is attached with $\delta(s, \alpha)$ (explored so far), and a pointer field, called *self_ptr*. The *self_ptr* points to the *h-segments* in the *HDFR*, that has introduced the point α in the MIN_HEAP. Execution starts by putting s in MIN_HEAP with $\delta(s, s) = 0$, and proceeds in a manner similar to Dijkstra's shortest path algorithm. But, unlike Dijkstra's algorithm, here the event-points are generated during the execution and are inserted in the MIN_HEAP. For each element in the MIN_HEAP, the monotone descent path from s exists, but there is a possibility of obtaining an alternate path of smaller length. The exception holds for the root node (say v) of the MIN_HEAP, whose attached δ-value cannot be reduced further by choosing an alternate monotone descent path.

Each time, we choose the member v in the MIN_HEAP having minimum δ-value. At the end of this subsection, we will prove that the δ-value attached to v is indeed $\delta(s, v)$ (length of the shortest monotone descent path $\pi(s, v)$). We *permanently mark* v, and process its adjacent face(s) to include some more region in $DFR(s)$. Let the *link-vertex* attached to v be u, and the *h-segment* attached to v be $[\alpha, \beta]$, where $[\alpha, \beta]$ is on an edge $e_0 = (a, b)$. We process the face $f = \Delta abc$ which is adjacent to e_0, and is in the other side of the vertex u. Let us name the other two edges of face f as $e_1 = (a, c)$, $e_2 = (b, c)$ respectively. We need to consider two distinct cases: (i) v is not a vertex of $DFR(s)$, (see Fig. 7) and (ii) v is a vertex of $DFR(s)$ (see Fig. 9).

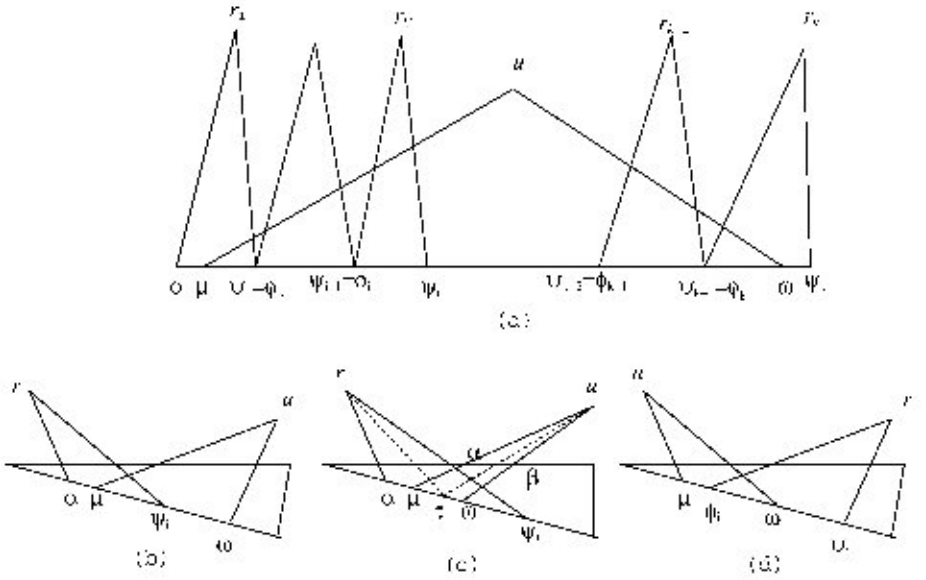

Fig. 8. Overlaps of I with other *h-segments*.

Case (i): v is not a vertex of $DFR(s)$

Let $\mathcal{R}$ be the planar unfolding of all the faces intersected by the path $\pi(u, v)$. We unfold face f onto $\mathcal{R}$, join (u, α) and (u, β) and extend these lines inside face f. These lines hit the boundary of f at μ and ω respectively. Let I be the portion of the boundary of f from μ to ω, which is straight-line reachable from u in the planar unfolding $\mathcal{R}$. I contributes one or two *h-segments* in the *HDFR* depending on whether it is a single interval (on either e_1 or e_2) or contains the vertex c (see Figs. 7(a) and 7(b) respectively). We compute $\delta_u(\mu)$ and $\delta_u(\omega)$ and search the interval $[\mu, \omega]$ in the $AVL(e)$ attached to the edge e in the *HDFR* data structure.

If I does not overlap with the existing *h-segments* then we (i) insert the *h-segment* $[\mu, \omega]$ in $AVL(e)$, (ii) insert μ and ω in the MIN_HEAP with respect to $\delta_u(\mu)$ and $\delta_u(\omega)$ respectively, (iii) insert the *frontier-point* $\phi \in [\mu, \omega]$ (with respect to $\delta_u(\phi)$) if it does not coincide with any of μ and ω, and (iv) set the *self_ptr* of μ, ω and ϕ to point $I = [\mu, \omega]$ in the *HDFR* data structure.

If I overlaps with the *h-segments* $\{J_i = [\phi_i, \psi_i],\ i = 1, \ldots, k\}$, $k \geqslant 1$ on an edge e, $\mu \in J_1$ and $\omega \in J_k$ (see Fig. 8(a)), and the *link-vertex* attached to J_i is r_i, then we consider each interval J_i, $i = 1, 2, \ldots, k$ in order. For each J_i, we compute $\delta_{r_i}(\phi_i)$, $\delta_{r_i}(\psi_i)$, $\delta_u(\phi_i)$ and $\delta_u(\psi_i)$. Depending on the relationship among these four quantities, we may have to replace J_i in $AVL(e)$ by some newly generated *h-segments* as described below. The same technique was followed in [13]. If $J_i = [\phi_i, \psi_i]$ needs to be replaced then we split it two or three pieces depending on the following situations:

Case 1. $\mu \in [\phi_i, \psi_i]$ but $\omega \notin [\phi_i, \psi_i]$: Here J_i splits into two pieces, namely $J_{1i} = [\phi_i, \mu]$ and $J_{2i} = [\mu, \psi_i]$ (see Fig. 8(b)).

Case 2. $\mu, \omega \in [\phi_i, \psi_i]$: Here J_i splits into three pieces, namely $J_{1i} = [\phi_i, \mu]$, $J_{2i} = [\mu, \omega]$ and $J_{3i} = [\omega, \psi_i]$ (see Fig. 8(c)).

Case 3. $\mu \notin [\phi_i, \psi_i]$ but $\omega \in [\phi_i, \psi_i]$: Here J_i splits into two pieces, namely $J_{2i} = [\phi_i, \omega]$ and $J_{3i} = [\omega, \psi_i]$ (see Fig. 8(d)).

In either of these situations, we identify the *tie-point* $\tau \in J_{2i}$ (see p. 658 of [13]), where $\delta_{r_i}(\tau) = \delta_u(\tau)$ (if it exists). If the *tie-point* (τ) is found, then it splits J_{2i} into two intervals. Finally, we delete J_i and insert all the newly generated

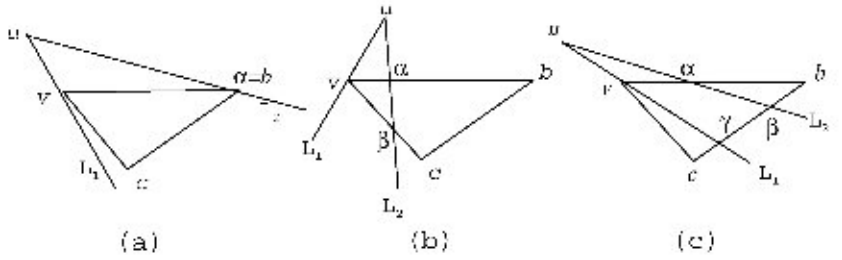

Fig. 9. Processing a vertex.

h-segments (with their corresponding predecessors and link-vertices) in the *HDFR* data structure, where

deletion of a *h-segment* from an edge e implies its removal from $AVL(e)$, and deletion of its two *break-points* and one *frontier-point* (if it exists) from MIN-HEAP. These three elements in the MIN-HEAP are accessed using ptr_1, ptr_2 and ptr_3 attached to that *h-segment*, and

insertion of a *h-segment* on an edge e implies its insertion in $AVL(e)$, setting the *link-vertex*, predecessor, and *self_ptr* of its two *break-points* and the *frontier-point* (if it is different from one of its *break-points*), and finally inserting these two/three elements in the MIN-HEAP. Finally, the ptr_1, ptr_2 and ptr_3 of the *h-segment* are set to point these three elements in the MIN-HEAP.

After processing all the J_is' for $i = 1, 2, \ldots, k$, we replace all the *h-segments* in $AVL(e)$ having the same link-vertex u by a single *h-segment* which is obtained by merging them.

If the above steps are executed for at least one J_i, then we insert μ (or the corresponding *tie-point*) and ω (or the corresponding *tie-point*) as break-points in the MIN_HEAP along with their respective δ-values and *self_ptr*. For each newly inserted *h-segment* the corresponding *frontier-point* is also to be inserted in MIN_HEAP.

Case (ii): v is a vertex of $DFR(s)$

Let the *h-segment* attached to v be $[v, \alpha]$. We unfold the face f onto $\mathcal{R}$, and extend the straight lines $L_1 = (u, v)$ and $L_2 = (u, \alpha)$ beyond v and α respectively. If both L_1 and L_2 go outside f then both the edges e_1 and e_2 of f are straight line reachable from u, and they will be considered as *h-segments* (see Fig. 9(a)). If one or both of L_1 and L_2 hit(s) the boundary of f, then one or two *h-segments* will be generated (see Figs. 9(b), (c)). For each of them, *link-vertex* is u and predecessor is $[v, \alpha]$.

In addition, we need to consider each edge e in $IDFR(v)$ (defined in Section 3) which are not adjacent to v. If e (or a portion of e) lies to the other side of α with respect to the line L_1, then it is considered as *h-segment* with *link-vertex* and predecessor both equal to v. Note that similar technique was adopted in [13], but the monotone descent property from v was not required there.

Each of these newly generated *h-segments* may overlap on some existing *h-segments*, and these are tackled using the same technique as mentioned in Case (i). Finally, all the newly generated *h-segment* are inserted in respective *AVL*-trees of the *HDFR* data structure, and all the break-points and a *frontier-point* are inserted in MIN_HEAP. If v is observed to be a break-point of a *h-segment*, then it is also to be inserted in MIN_HEAP for processing the other faces incident to v.

Role of the frontier-points

Consider the planar unfolding along an edge sequence of the terrain $\mathcal{T}$. The *break-points* a, b, c and d are stored in the MIN_HEAP and their distances are shown inside square brackets in Fig. 10. If we do not consider the *frontier-point* then after processing d from MIN_HEAP, the point q will be pushed to MIN_HEAP, with $\delta(s, q) = \delta_{r_2}(q) = 39$ (say). As $\delta(s, q) < \min(\delta(s, a), \delta(s, b))$, after processing c from MIN_HEAP, q will be chosen for processing. This implies, the distance of q will not be reduced further by Lemma 8. But Fig. 10 shows that q is also straight-line reachable from r_1 and $\delta_{r_1}(q) = 38$. Note that if we consider *frontier-points* γ_1 and γ_2, then $\delta_{r_2}(q) > \min(\delta(s, \gamma_2), \delta(s, \gamma_1))$. Thus, either γ_1 or γ_2 will be chosen prior to the processing of q, and $\delta(s, q)$ will be set correctly prior to its processing as described in the proof of following lemma.

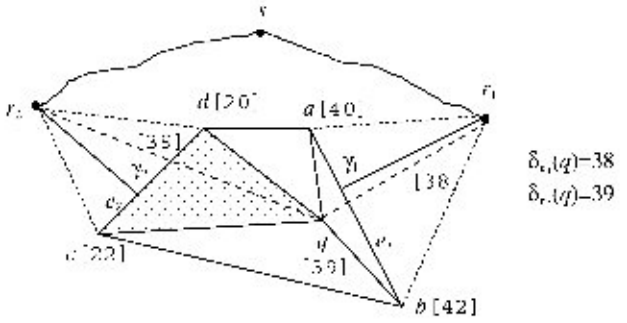

Fig. 10. Demonstration of frontier point.

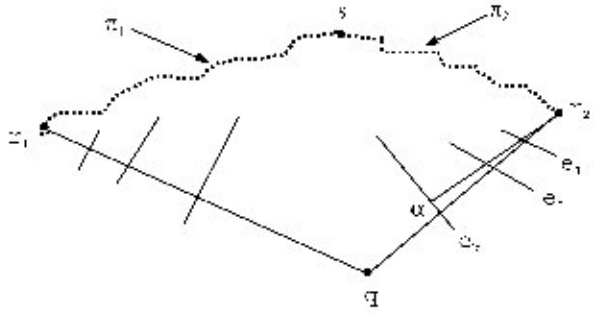

Fig. 11. Proof of Lemma 8.

Lemma 8. *Every time the path-length attached to the top-most element of the MIN_HEAP is optimum.*

Proof. [By contradiction] Let the distance attached to q in the MIN_HEAP represents the length of the path π_1 from s to q through the *link-vertex* r_1, and through the edge sequence $\mathcal{E}_1$ (see Fig. 11). Suppose π_1 is not optimum; there exists another path π_2 from s to q through the *link-vertex* r_2 and passing through the edge sequence $\mathcal{E}_2$, such that $\delta_{r_1}(q) > \delta_{r_2}(q)$. Let $e_k \in \mathcal{E}_2$, and is closest to q. Since $\delta_{r_2}(r_2) < \delta_{r_2}(q) < \delta_{r_1}(q)$, and q is currently being processed. Moreover, since π_2 passes through e_k, there exists a frontier point, say α, on e_k with respect to r_2, and $\delta_{r_2}(\alpha) < \delta_{r_2}(q) < \delta_{r_1}(q)$. While processing α, we will discover q, and compute $\delta_{r_2}(q)$. Thus, the hypothesis that distance attached to $q = \delta_{r_1}(q)$ is not correct. □

Lemma 9. *In Phase* 1, *the h-segments on all the edges of* $\mathcal{T}$ *are computed correctly.*

Proof. A *h-segment* I, generated in Phase 1, is determined by its two *break-points* which are permanently marked. By Lemma 8, for each of these *break-points* v, $\delta(s, v)$ is correctly computed. The lemma holds for each interior point of I, because of the fact that *h-segments* are non-overlapping and for each point $\alpha \in I$, $\pi(s, \alpha)$ passes through the *link-vertex* attached to I, which is also permanently marked. □

Phase 2. In this phase, we compute the homogeneous partition inside each face separately as was done in [13]. Let us consider the *h-segments* on all the edges of a face $f = \Delta abc$, and their corresponding *link-vertices*. Each *link-vertex* v is attached with an additive weight $\delta(s, v)$. We unfold the faces of the terrain such that the *link-vertices* of all the *h-segments* belong to the same plane containing f. Next, we compute the Voronoi diagram of these (additive) weighted points [7]. The portion of this diagram inside the face Δabc is the homogeneous partition $VOR(f)$ of face f. The computation of homogeneous partitions inside a face needs $O(K \log K)$ time, where K is the number of *h-segments* on the boundary of Δabc. Each such partition points to its corresponding *h-segment* in the *HDFR* data structure.

4.2. Query answering

For a given query point t, we first locate the face $f = \Delta abc$ of $DFR(s)$ containing t. Next, we need to search in the data structure $VOR(f)$ to decide the appropriate cell inside f in which the query point lies. This actually gives the *h-segment* I through which the shortest path from s to t has entered that face, and the length of the corresponding path (see Theorem 3). We use predecessor links to obtain the edge sequence $\mathcal{E}^* = \{e_1^*, e_2^*, \ldots, e_m^*\}$, to reach the *link-vertex* r (a vertex of $DFR(s)$) attached to I. Thus, the shortest monotone descent path $\pi(r,t)$ passes through faces $f_0^*, f_1^*, f_2^*, \ldots, f_m^* = f$, where r is a vertex of f_0^*, and $t \in f_m^*$, and the edge e_i^* is shared by f_{i-1}^* and f_i^*. Finally, we obtain the entire path $\pi(s,t)$ using the following steps.

- Compute $\pi(r,t)$ through the edge sequence $\mathcal{E}^*$ as mentioned above.
- From r, reach its *link-vertex* r_1 through an edge sequence obtained using predecessor pointers. Compute the planar unfolding of the faces, adjacent to that edge sequence. The inverse image of the line segment $[r_1, r]$ in the unfolded plane is the shortest descent flow path from r_1 to r.
- Treat r_1 as r, and repeat step (ii) until the vertex s is reached.

4.3. Complexity analysis

The total number of vertices and edges in $DFR(s)$ are both $O(n)$. Let us consider the *h-segments* on an edge e coming through one of its adjacent faces. Each such *h-segment* is designated by two lines originating from its link vertex and is supported by some other vertex of $\mathcal{T}$. Note that one vertex cannot support more than one such lines. Thus, the number of *h-segments* on an edge of $\mathcal{T}$ is $O(n)$ in the worst case. The final set of *h-segments* on an edge is obtained by merging the two sets of *h-segments* coming through its two adjacent faces, which is also $O(n)$. Each *h-segment* is attached with a *frontier-point*. Thus, the total number of event-points pushed in the MIN_HEAP may be $O(n^2)$ in the worst case. Processing of all these event-points needs $O(n^2 \log n)$ time. Finally, in Phase 2, the time complexity of computing the homogeneous partitioning of all the faces is $O(n^2 \log n)$ in total, and it produces $O(n^2)$ homogeneous partitions. The worst case size of the *HDFR* is $O(n^2)$.

The query time needs $O(\log n + k)$, where $O(\log n)$ time is required for the point location queries with the query point t as mentioned in Subsection 4.2, and k is the number of line segments on the shortest path from s to t. Thus we have the following theorem:

Theorem 2. *Given a convex $DFR(s)$ of a polyhedral terrain $\mathcal{T}$ with n vertices, and a source point s, our algorithm* (i) *creates the HDFR data structure in $O(n^2 \log n)$ time and $O(n^2)$ space.* (ii) *For a given query point $t \in DFR(s)$, it outputs a monotone descent path from s to t in $O(k + \log n)$ time, where k is the number of line segments on the optimal path.*

4.4. A simple variation: the distance query

A simpler version of the above problem is the *distance query*, where the objective is to compute the length of the monotone descent path from s to a given query point t. We show that a minor tailoring of the *HDFR* data structure helps us to report the length in $O(\log n)$ time.

Recall the Phase 2 of the preprocessing. Here, we have assumed that the length of the shortest path of each *link-vertex* is already known. At each face, we have considered the *link-vertices* attached to the *h-segments* on its boundary as weighted points, where the weight attached to a link vertex is the length of the shortest monotone descent path from s to that point. Next, we computed the Voronoi diagram of those weighted points inside that face. At each partition, we attach two scalar information: (i) the coordinate of the image of its corresponding *link-vertex* in the unfolded plane, and (ii) the distance of that *link-vertex* from s.

During the distance query for a query point t, we identify the partition in which it belongs by point location. Let r be the *link-vertex* attached to that partition. The length of the shortest monotone path from s to t is obtained by $\delta(s,r) + dist(r,t)$, where $dist(r,t)$ is basically the Euclidean distance of this point from the image of the *link-vertex* r in the unfolded plane. Thus, we have the following result:

Theorem 3. *Given a polyhedral terrain $\mathcal{T}$ with n vertices, and a source point s, the revised HDFR data structure can be created in $O(n^2 \log n)$ time and $O(n^2)$ space, such that given any arbitrary query point t, the distance query can be answered in $O(\log n)$ time.*

5. Shortest monotone descent path through parallel edge sequence

In this section, we shall consider a slightly different problem on a general terrain where each pair of adjacent faces are not restricted to only in convex position. Here, along with the source (s) and destination (t) points, a sequence of faces $\mathcal{F} = \{f_0, f_1, \ldots, f_m\}$, $s \in f_0$, $t \in f_m$, is given. The objective is to find the shortest descent flow path through $\mathcal{F}$. This problem in its general form seems difficult. But we are proposing an efficient solution in a restricted setup. Let $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$ be the sequence of edges separating the consecutive faces in $\mathcal{F}$. We assume that the members in the edge sequence $\mathcal{E}$ are parallel to each other. Note that here we are deviating from the assumption that the faces in the terrain are triangular.

The problem is very much similar to the *L_2-shortest path problem over walls*, defined in [15]. Here a set of n vertical walls parallel to the x-axis are given. Each wall is positioned on the xy-plane. The ith wall, is positioned at $y = a_i$, and its top boundary, denoted by e_i, is a line of the form $z = b_i x + c_i$, where a_i, b_i and c_i are given constants, $a_1 < a_2 < \cdots < a_n$. The objective is to report the Euclidean shortest path between a given pair of query points s and t, where $y(s) < a_1$ and $y(t) > a_n$. They proved that the shortest path is always monotone with respect to the y-axis, and it bends on the edges e_i. It is also proved that the shortest path from s to t is the concatenation of two sub-paths, one of them is monotone ascending and the second one is monotone descending with respect to z-coordinate. Our case is a simpler version, where the plane between two consecutive edges is known, and we are specifically searching for a monotone descent path which is constrained to lie on the plane.

5.1. Properties of parallel edge sequence

Lemma 10. *Let p and q be two points on two consecutive members e_i and e_{i+1} of $\mathcal{E}$ which bounds a face f, and $z(p) = z(q)$. Now, if a line ℓ on face f intersects both e_i and e_{i+1}, and is parallel to the line segment $[p, q]$, then* (i) *the length of the portion of ℓ lying in face f is equal to the length of the line segment $[p, q]$, and* (ii) *all the points on ℓ have the same z-coordinate.*

Proof. Part (i) of the lemma follows from the fact that as e_i and e_{i+1} are parallel, the portion of ℓ in face f and the line segment $[p, q]$ appear as two parallel edges of a parallelogram on face f.

Part (ii) of the lemma trivially follows if the face f is horizontal. So, we prove it for the case where f is not horizontal.

Consider a horizontal plane h at altitude $z(p)$. The intersection of the face f and the plane h is the line segment $[p, q]$ (by part (i) of this lemma). Consider another horizontal plane h' through a point r on line ℓ. The intersection of f and h' must be parallel to $[p, q]$, and hence it coincides with the line ℓ. Thus, all the points on the line ℓ have the same z-coordinate. □

Lemma 11. *Let e_i and e_{i+1} be two edges in $\mathcal{E}$ bounding a face f. For a pair of points p, $p' \in e_i$ and a pair of points $q, q' \in e_{i+1}$, if $z(p) > z(p')$ and $z(q) > z(q')$, then the line segments $[p, q]$ and $[p', q']$ does not intersect in face f; but the line segments $[p, q']$ and $[p', q]$ must intersect in face f.*

Proof. Without loss of generality, assume that $z(p) > z(q)$. Draw two horizontal planes h_1 and h_2 through p and q respectively which intersect face f along the lines ℓ_1 and ℓ_2 respectively. Note that ℓ_1 is above ℓ_2 with respect to their z-coordinates. Here any one of the three cases may arise: (i) both p' and q' are above h_2, (ii) both p' and q' are below h_2, (iii) p' and q' appear in different sides of h_2. Case (i) is impossible since $z(q) > z(q')$. In Case (ii), the line segment $[p, q]$ and $[p', q']$ appear in different sides of the plane h_2, and hence they cannot intersect (see Fig. 12(a)). In Case (iii), $z(p') > z(q)$ and $z(q') < z(q)$. Here, if $[p, q]$ and $[p', q']$ intersects, then e_i and e_{i+1} cannot be parallel (see Fig. 12(b)). The reason is that, as e_i and e_{i+1} are parallel and $z(p) > z(p') > z(q)$ & $z(q) > z(q')$, then $[p, q]$ and $[p', q']$ are the sides (not the diagonals) of the trapezoid $\square pp'q'q$ (see Fig. 12(c)). □

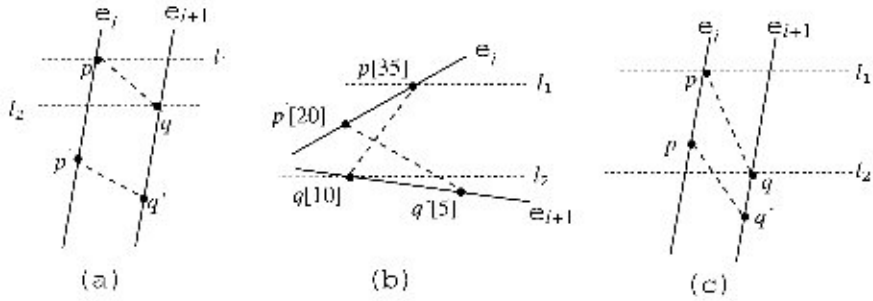

Fig. 12. Proof of Lemma 11.

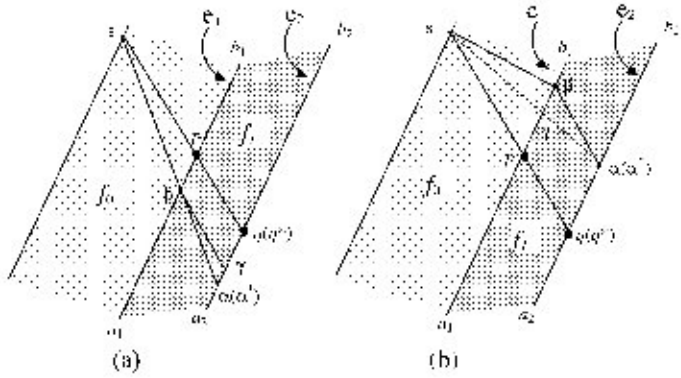

Fig. 13. Proof of Theorem 4.

Theorem 4. *Let f_1 be a non-horizontal face bounded by two parallel edges $e_1 = [a_1, b_1]$ and $e_2 = [a_2, b_2]$ $(z(a_i) < z(b_i)$, $i = 1, 2)$; the point s appears in its adjacent face f_0 such that f_0 and f_1 are separated by the edge e_1. If there exist a pair of points $p \in e_1$ and $q \in e_2$ with $z(p) = z(q) < z(s)$, and the points s, p, q^* (q^* is the image of the point q in the planar unfolding $U(e_1)$) are collinear, then*

(i) *for any point α in the interval $[q, a_2]$, the shortest monotone descent path along e_1 is the inverse-image of the straight line segment $[s, \alpha^*]$ in the unfolded plane provided $[s, \alpha^*]$ intersects the edge e_1 in its interior;*
(ii) *for any point α in the interval $[b_2, q]$, the shortest monotone descent path along e_1 is not an inverse-image of the straight line segment $[s, \alpha^*]$ in unfolded plane. Here $\pi_{\mathrm{md}}(s, \alpha)$ will pass through a point $\beta \in [b_1, p]$ with $z(\beta) = z(\alpha)$ in the original terrain.*

Proof. The line segment $[p, q]$ partitions the face f_1 into two parts, and the points b_1 and b_2 belong to the same side of $[p, q]$ (by Lemma 11). Consider a point $\alpha \in [q, a_2]$ on the edge e_2 (see Fig. 13(a)). In the planar unfolding $U(e_1)$, the straight line segment $[s, \alpha^*]$ intersects e_1 at a point, say β. By Lemma 11, the line segment $[\alpha, \beta]$ is below the line segment $[p, q]$. Thus, if β is in the interior of the edge e_1 then $\beta \in [p, a_1]$. Let us consider a line segment $[\beta, \gamma]$ on face f_1 which is parallel to $[p, q]$, and γ is on the edge e_2. Now consider the triangle $\Delta s q^* \alpha^*$ in the unfolded plane, where the point β lies on $[s, \alpha^*]$. As the line segment $[\beta, \gamma^*]$ is parallel to $[s, q^*]$, γ lies on $[q^*, \alpha^*]$. So, $z(\alpha) < z(\gamma) < z(q)$. By Lemma 10, $z(\gamma) = z(\beta)$. Hence part (i) of the lemma follows.

The proof of part (ii) follows from the following argument. Consider a point $\alpha \in [q, b_2]$ (see Fig. 13(b)); the line segment $[s, \alpha^*]$ intersects the edge e_1 at γ in the unfolded plane $U(e_1)$. Draw a line segment $[\alpha, \beta]$ on face f_1 which is parallel to $[p, q]$. As $z(\beta) = z(\alpha)$ (by Lemma 10), we have $z(\gamma) < z(\alpha)$. Thus, the shortest monotone descent path from s to α cannot be the geodesic shortest path between them. As $z(\beta) = z(\alpha) < z(s)$, the shortest monotone descent path from s to α will be the concatenation of line segments $[s, \beta]$ and $[\beta, \alpha]$. □

In order to describe our algorithm, let us introduce the following terminology.

We obtain the planar unfolding of the faces $\mathcal{F} = \{f_1, f_2, \ldots, f_m\}$ onto face f_0, and use a two-dimensional coordinate system for the entire unfolded plane such that the members in the edge sequence $\mathcal{E} = \{e_1, e_2, \ldots, e_m\}$ are ordered

from left to right, and each of them is parallel to the y-axis. The z-coordinate of a point in the unfolded plane indicates the z-coordinate of that point in the original terrain. In this planar unfolding, if an edge e_i of the terrain is represented as $[a_i, b_i]$, with $y(a_i) < y(b_i)$ then $z(a_i) \leqslant z(b_i)$ (see Lemma 10). The source s is in f_0, then flow passes through the edge sequence $\mathcal{E}$ to reach a point $t \in f_m$. If a path $\pi(s,t)$ enters into a face f_i along a line ℓ_i, then the *angle of incidence* of $\pi(s,t)$ in face f_i (with edge e_i) is denoted by θ_i, and henceforth will be referred to as *slope* of ℓ_i.

Let e_1 and e_2 be two parallel boundaries of a face f. The *translation event* for face f, denoted by $T(f)$ is a linear translation of e_2 on e_1 such that the entire face f is merged to the line e_1 as follows:

The points in the unfolded plane lying on the same side of s with respect to e_1 remain unchanged.

Each point p lying in the proper interior of the face f is mapped to a point $q \in e_1$ such that $z(p) = z(q)$.

Each point $p = (x(p), y(p))$ on the edge e_2 is mapped to a point $q = (x(q), y(q))$ on the edge e_1 such that $z(p) = z(q)$. Under this transformation $x(q) = x(p) + \alpha$, $y(q) = y(p) + \beta$, where the tuple (α, β) are constant, and they depend on the slope and width of the face f.

Each point (x, y) in the unfolded plane lying on the other side of s with respect to e_2 is moved to the point $(x + \alpha, y + \beta)$.

The slope of the line containing $[p, q]$ is referred to as *merging direction* of face f, and is denoted as $\phi(f)$. Theorem 4 indicates the following result.

Corollary 4.1. *If the slope θ of a line segment ℓ in face f is such that* (i) $\theta < \phi(f)$ *then ℓ is strictly monotone descent,* (ii) $\theta = \phi(f)$ *then all the points in ℓ have same z-coordinate, and* (iii) $\theta > \phi(f)$ *then ℓ is strictly monotone ascent.*

Let $\pi_{\mathrm{md}}(s,t)$ be the shortest monotone descent path from $s \in f_0$ to $t \in f_m$ passing through a sequence of parallel edges $\{e_1, e_2, \ldots, e_{m-1}\}$. Along this path there exists a set of faces $\{f_{j_i}, i = 1, 2, \ldots, k\}$ such that all the points of the path $\pi_{\mathrm{md}}(s,t)$ in face f_{j_i} have same z-coordinate ξ_{j_i}; the portions of the path in all other faces are strictly monotone descent. Now, we have the following theorem.

Theorem 5. *If the translations $T(f_{j_1}), T(f_{j_2}), \ldots, T(f_{j_k})$ are applied (in any order) on the unfolded plane of faces $f_0, f_1, \ldots, f_m$ then the shortest monotone descent path $\pi_{\mathrm{md}}(s,t)$ will become a straight line segment from s to t in the transformed plane.*

Proof. Let us first assume that $k = 1$, i.e., $\pi_{\mathrm{md}}(s,t)$ passes through a face f with all points having the same z-coordinate. Let f_a and f_b be its preceding and succeeding faces with separating edges e_a and e_b respectively. We also assume that $\pi_{\mathrm{md}}(s,t)$ consists of three consecutive line segments $[s,a]$, $[a,b]$, $[b,t]$ lying in f_a, f and f_b respectively. Note that all the points on $[a,b]$ have same z-coordinate. If we apply $T(f)$, the points b and t will be mapped to a and t'. Now, in the transformed plane, the shortest path from s to t' is the straight line segment $[s,t']$. We argue that $[s,t']$ will pass through a. On the contrary, assume that $[s,t']$ intersect e_a at a', and a' is the image of $b' \in e_b$ under $T(f)$, $b' \neq b$. Thus, $d(s,a') + d(a',t') < d(s,a) + d(a,t')$. Now, applying reverse transformation, $d(s,a') + d(b',t) < d(s,a) + d(b,t)$. From Lemma 10, $d(s,a') + d(a',b') + d(b',t) < d(s,a) + d(a,b) + d(b,t)$. This leads to a contradiction.

Let there exist several faces on the path $\pi_{\mathrm{md}}(s,t)$ such that all the points of $\pi_{\mathrm{md}}(s,t)$ in that face have same z-coordinate. If we apply the transformation T on one face at a time, the above result holds. The order of choosing the face for applying the transformation T is not important due to the following argument: (i) a point p on the unfolded plane will be affected due to same set of transformation irrespective of in which order they are applied, and (ii) the effects of all the transformations affecting on a point are additive. □

Lemma 12. *If the shortest monotone descent path $\pi_{\mathrm{md}}(s,t)$ is passing through a sequence of parallel edges, then all the line segments of $\pi_{\mathrm{md}}(s,t)$, which are strictly monotone descent, are parallel on the unfolded plane of all faces.*

Theorem 6. *If the line segments of shortest monotone descent path $\pi_{md}(s,t)$ in faces $f_{1^*}, f_{2^*}, \ldots, f_{k^*}$ are strictly monotone then their slopes are equal. The slope of the portions of $\pi_{md}(s,t)$ in all other faces are equal to the merging angle of the corresponding faces.*

The above discussions lead to the following algorithm for computing the shortest monotone descent path between a pair of points s (source) and t (destination) through a sequence of faces separated by mutually parallel edges.

5.2. Algorithm

Step 1. We compute the planar unfolding where the faces $f_1, f_2, \ldots, f_m$ are unfolded onto face f_0 containing s. We assume that the entire terrain is in first quadrant, and all the edges of $\mathcal{T}$ are parallel to the y-axis.

Step 2. We compute the merging angle for all the faces $f_i, i = 1, 2, \ldots, m$, and store them in an array Φ in ascending order. Each element contains its face-id.

Step 3. (*Merging phase*) Let θ be the slope of the line joining s and t in the unfolded plane. We sequentially inspect the elements of the array Φ from its first element onwards until an element $\Phi[k] > \theta$ is obtained. For each element $\Phi[i], i < k$, the translation event takes place, and we do the following:

Let $\Phi[i]$ correspond to a face f. We transform the entire terrain by merging the two boundaries of face f, i.e., compute the destination point t under the translation. The face f is marked. We update θ by joining s with the new position of t.

Compute the optimum path in transformed plane by the line joining s and t.

Step 4. After the execution of Step 3, the value of θ indicates the slope of the path segments which are strictly monotone descent along $\pi_{md}(s,t)$. We compute $\pi_{md}(s,t)$ as follows:

Start from the point s at face f_0, and consider each face $f_i, i = 1, 2, \ldots, m$ in order. If face f_i is not marked, $\pi_{md}(s,t)$ moves in that face along a line segment of slope θ; otherwise, $\pi_{md}(s,t)$ moves along a line segment of slope $\Phi[i]$.

Step 5. Finally, report the optimum path $\pi_{md}(s,t)$.

5.3. Correctness and complexity analysis of the algorithm

Theorem 7. *Our algorithm correctly computes the shortest monotone descent path between two query points s and t through a sequence of faces of a polyhedral terrain bounded by parallel edges in $O(m \log m)$ time.*

Proof. We prove the correctness of the algorithm by contradiction. The path obtained by our algorithm is $\pi(s,t)$. It passes through the faces at equal altitude for which the merging angles are $\{\Phi[1], \ldots, \Phi[k]\}$ $(\subseteq \Phi)$, and follows strictly monotone descent (with angle $= \theta$) in the faces having merging angles $\{\Phi[k+1], \Phi[k+2], \ldots, \Phi[m]\}$, where $\Phi[i] < \theta$ for $i = 1, \ldots, k$, and $\Phi[i] > \theta$ for $i = k+1, \ldots, m$.

Let the optimum path $\pi'(s,t)$ passes through the faces at equal altitude for which the merging angles are $\{\Phi[1], \ldots, \Phi[k']\}$ $(\subseteq \Phi)$, and follows strictly monotone descent (with angle $= \theta'$) in the faces having merging angles $\{\Phi[k'+1], \Phi[k'+2], \ldots, \Phi[m]\}$, where $\Phi[i] < \theta'$ for $i = 1, \ldots, k'$, and $\Phi[i] > \theta'$ for $i = k'+1, \ldots, m$.

If $k = k'$, then $\theta = \theta'$ (by Theorems 5 and 6), and the path obtained by our algorithm is optimum.

Let us assume that $k < k'$, or in other words, $\theta < \theta'$ (since Φ is created in ascending order of merging angles). Thus, our algorithm chooses few more faces than the optimum solution where the path goes through the same height. Now, the three cases stated below are exhaustive.

As $\theta < \theta'$, $\pi'(s,t)$ will diverge upwards from $\pi(s,t)$ in all the faces where both the paths follow strictly monotone descent property.

In some faces $\pi'(s,t)$ goes through equal height but $\pi(s,t)$ follows monotone descent property. There also $\pi'(s,t)$ will diverge upwards from $\pi(s,t)$.

In those faces where $\pi'(s,t)$ and $\pi(s,t)$ goes through equal height, they remain parallel.

Since $\pi(s,t)$ has reached t, $\pi'(s,t)$ will reach somewhere above t in face f_m. The similar argument proves that $k \neq k'$.

Given a sequence of m faces of a polyhedral terrain bounded by parallel lines, and two query points s and t, Steps 1 and 2 of the algorithm computes the merging angles and sorts them in $O(m \log m)$ time. Step 3 needs $O(1)$ time. Each iteration of Step 4 needs $O(1)$ time, and we may need $O(n)$ such iterations for reporting the shortest monotone descent path from s and t. Thus the time complexity result follows. □

6. Conclusion

We have proposed polynomial time algorithms for finding the shortest monotone descent path from a point s to a point t in a polyhedral terrains in two special cases where (i) t is a point in convex $DFR(s)$, and (ii) the path from s to t passes through a set of faces bounded by parallel edges. The same problem for the general terrain is still unsolved.

Acknowledgements

We sincerely acknowledge the two anonymous referees of this paper. Their suggestions helped us enormously to improve the quality of the paper.

References

[1] P.K. Agarwal, S. Har-Peled, M. Karia, Computing approximate shortest paths on convex polytopes, Algorithmica 33 (2002) 227–242.
[2] L. Aleksandrov, A. Maheshwari, J.-R. Sack, Approximation algorithms for geometric shortest path problems, in: Proc. Symp. on Theory of Comput., 2000, pp. 286–295.
[3] L. Aleksandrov, A. Maheshwari, J.-R. Sack, An improved approximation algorithms for computing geometric shortest paths problems, in: Proc. Symp. on Foundations of Computing Theory, 2003, pp. 246–257.
[4] L. Aleksandrov, A. Maheshwari, J.-R. Sack, Determining approximate shortest paths on weighted polyhedral surfaces, J. ACM 52 (2005) 25–53.
[5] M. de Berg, M. van Kreveld, Trekking in the Alps without freezing or getting tired, Algorithmica 18 (1997) 306–323.
[6] J. Chen, Y. Han, Shortest paths on a polyhedron, Internat. J. Comput. Geom. Appl. 6 (1996) 127–144.
[7] S. Fortune, A sweepline algorithm for Voronoi diagrams, Algorithmica 2 (1987) 153–174.
[8] S. Kapoor, Efficient computation of geodesic shortest paths, in: Symp. on Theory of Computing, 1999, pp. 770–779.
[9] J. Hershberger, S. Suri, Practical methods for approximating shortest paths on a convex polytopes in R^3, Comput. Geom. Theory Appl. 10 (1998) 31–46.
[10] D.G. Kirkpatrick, Optimal search in planar subdivisions, SIAM J. Comput. 12 (1983) 28–35.
[11] M. Lanthier, A. Maheswari, J.-R. Sack, Approximating weighted shortest paths on polyhedral surfaces, Algorithmica 30 (2001) 527–562.
[12] C. Mata, J.S.B. Mitchell, A new algorithm for computing shortest paths in weighted planar subdivisions, in: Proc. 13th ACM Symp. Comput. Geom., 1997, pp. 264–273.
[13] J.S.B. Mitchell, D.M. Mount, C.H. Papadimitrou, The discrete geodesic problem, SIAM J. Comput. 16 (1987) 647–668.
[14] J.S.B. Mitchell, C.H. Papadimitrou, The weighted region problem: Finding shortest paths through a weighted planar subdivision, J. Assoc. Comput. Mach. 38 (1991) 18–73.
[15] J.S.B. Mitchell, M. Sharir, New results on shortest paths in three dimensions, in: Proc. of the Twentieth Symposium on Computational Geometry, 2004, pp. 124–133.
[16] J.H. Reif, Z. Sun, An efficient approximation algorithm for weighted region shortest path problem, in: Proc. of the Workshop on Algorithmic Foundations of Robotics, 2000, pp. 191–203.
[17] S. Roy, S. Das, S.C. Nandy, A practical algorithm for approximating shortest weighted path between a pair of points on polyhedral surface, in: Int. Conf. on Computational Science and Its Applications, 2004, pp. 42–52.
[18] J.R. Sack, J. Urrutia, Handbook of Computational Geometry, North-Holland/Elsevier, Netherlands, 2000.
[19] M. Sharir, A. Schorr, On shortest paths in polyhedral space, SIAM J. Comput. 15 (1986) 193–215.
[20] K.R. Varadarajan, P.K. Agarwal, Approximating shortest paths on a non-convex polyhedron, SIAM J. Comput. 30 (2001) 1321–1340.