# Movement Planning in the Presence of Flows [*]

John Reif[†] Zheng Sun[‡]

**Abstract**

This paper investigates the problem of time-optimum movement planning in two and three dimensions for a point robot which has bounded control velocity through a set of $n$ polygonal regions of given translational flow velocities. This intriguing geometric problem has immediate applications to macro-scale motion planning for ships, submarines and airplanes in the presence of significant flows of water or air. Also, it is a central motion planning problem for many of the meso-scale and micro-scale robots that recently have been constructed, that have environments with significant flows that affect their movement. In spite of these applications, there is very little literature on this problem, and prior work provided neither an upper bound on its computational complexity nor even a decision algorithm. It can easily be seen that an optimum path for the $2\mathcal{D}$ version of this problem can consist of at least an exponential number of distinct segments through flow regions. We provide the first known computational complexity hardness result for the $3\mathcal{D}$ version of this problem; we show the problem is PSPACE hard. We give the first known decision algorithm for the $2\mathcal{D}$ flow path problem, but this decision algorithm has very high computational complexity. We also give the first known efficient approximation algorithms with bounded error.

## 1 Introduction

### 1.1 Formulation of the Problem and Motivation

We assume that the problem is given as a polyhedral decomposition of a two or three dimensional space, where each region $r$ defined by the polyhedral decomposition has an assigned *translational flow* defined by a vector $\overrightarrow{f_r}$. Each region $r$ is also associated with a non-negative real number $b_r$ giving the maximum Euclidean norm of the control velocity that the robot can apply within $r$. In particular, if the robot is traveling on the shared boundary between two regions, it can be considered as traveling inside either region, whichever is more favorable; the robot can always move by an infinitesimal distance into that region, and then travel inside that region along the boundary before eventually moving back onto the boundary. We define $\rho_r$ to be the ratio between $b_r$ and $|\overrightarrow{f_r}|$.

The robot is considered to be a point with a given initial position and also a given final position to be reached by the robot. At time $\tau = 0$, the point robot is at the given initial position point. Within each region $r$, the robot can apply, at each time $\tau \geq 0$ and in any direction, a translational *control velocity vector* $\overrightarrow{v(\tau)}$ of bounded Euclidean norm $|\overrightarrow{v(\tau)}| \leq b_r$. However, the *actual velocity*

[*]A preliminary version of this paper appeared in *Proceedings of the 7th International Workshop on Algorithms and Data Structures* (WADS2001), volume 2125 of *Lecture Notes in Computer Science.* This work was supported in part by DARPA/AFSOR Contract F30602-01-2-0561, NSF SEGR Grant NSF-11S-01-94604, NSF ITR Grant EIA-0086015, DARPA/NSF Grant CCR-9725021.

[†]Department of Computer Science, Duke University, Box 90129, Durham, NC 27708-0129, USA. reif@cs.duke.edu

[‡]Department of Computer Science, Hong Kong Baptist University, Kowloon, Hong Kong. sunz@comp.hkbu.edu.hk.

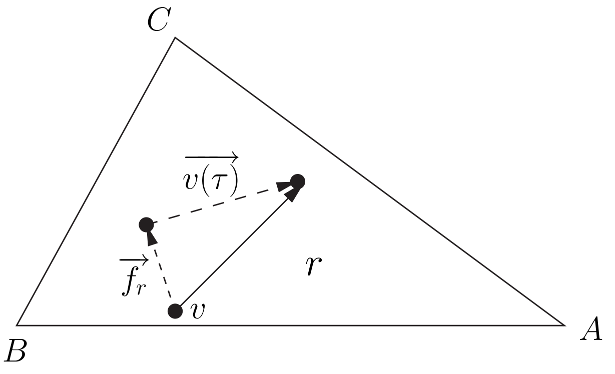


Figure 1: Composite velocity

of the robot at time $\tau$ is given by the sum $\overrightarrow{v(\tau)} + \overrightarrow{f_r}$ of its control velocity vector $\overrightarrow{v(\tau)}$ and the translational flow velocity $\overrightarrow{f_r}$ of region $r$, as shown by Figure 1.

The *flow path optimization problem* is to find an *optimum path* of movement of the point robot from the initial position to the final position with minimum time duration. The *flow path decision problem* is to determine if the flow path optimization problem has time $\tau_0$ for a given rational number $\tau_0 > 0$.

In an extreme example, where $b_r = 0$ at each region $r$, the movement of the robot is simply a sequence of translations provided by each region's flow, and the problem reduces to the prediction of the path of the robot in the presence of overwhelming flow velocities where the robot has no control of movement. In an another extreme example, where $|\overrightarrow{f_r}| = 0$ for each region $r$, this problem reduces to the usual weighted region optimum path problem through regions of given translational velocities.

In our investigation of the computational complexity of this problem, we assume that the polygonal decomposition of input problem has $n$ regions, and that the input problem is specified with a total of $n^{O(1)}$ bits: in particular, for some constant $c \geq 1$, we assume that we are given the following within $cn$ bits:

- the positions of the boundaries of these regions;
- the initial and final positions of the robot;
- the flow velocity and bounding velocity of the robot within each region.

This flow path problem has a number of macro-scale movement planning applications (where the size of the robot is about one centimeter or above):

- the problem of moving a ship on the surface of an ocean or river through regions where the surface currents have known flow velocity;
- the problem of moving a submarine through regions where the underwater currents have known flow velocity;
- the problem of moving an aircraft through regions where the air currents have known flow velocity.

The flow path problem becomes particularly relevant to these practical problems in the cases where the object to be moved is under autonomous control, and where the flow velocities are significant to require careful motion planning. This is an increasing occurrence as new robotic devices are developed that are of a rapidly decreasing size, and hence these meso and micro-scale robots can be strongly influenced by the local flows in their environment.

### 1.2 Previous Work and Our Results

Papadakis and Perakis [12, 13] previously gave heuristic algorithms for related problems such as for minimal time vessel routing in ocean currents. Sellen [21] studied the optimum route problem for a sailboat in a single region with multiple obstacles, where the velocity is a continuous function of sailing direction. There seems not to have been much other previous research on this problem, but there is considerable previous research on related movement problems.

Reif [15] provided the first PSPACE hardness result for a robotic motion planning problem, and Schwartz and Sharir [20] gave motion planning algorithms using the theory of real closed fields (Canny [4]). Reif and Sharir [16] gave algorithms and computational complexity lower bound results for robotic motion with moving obstacles (also see Wilfong [25]). Reif and Sun [19] showed that robotic motion planning in the presence of friction is undecidable.

Canny and Reif [5] showed the $3\mathcal{D}$ shortest path problem with polygonal obstacles is NP hard, and Reif and Storer [17] applied the theory of real closed fields to give a decision algorithm for this problem. The reference [10] surveys work on various $2\mathcal{D}$ and $3\mathcal{D}$ minimal cost path problems, including a variety of approximation algorithms for the weighted region optimum path problem given by Mitchell and Papadimitriou [11], Mata and Mitchell [9], Lanthier *et al.* [7], Aleksandrov *et al.* [1]. More recent works include Reif and Sun [18, 22, 23], and Aleksandrov *et al.* [2, 3] for the weighted region optimum path problem, as well as Sun and Reif [24], and Lanthier *et al.* [8] for the anisotropic optimum path problem.

One common approach for approximately solving these minimal cost path problems is to discretize the continuous geometric space by inserting discrete points (called *Steiner points*) on boundary edges. This approach has been applied to the $3\mathcal{D}$ Euclidean shortest path problem ([14]), the weighted region optimum path problem ([7, 1, 2, 18, 22, 23]), as well as the anisotropic optimum path problem ([8, 24]). In particular, Aleksandrov *et al.* [1, 2] proposed a logarithmic discretization scheme to compute an *$\epsilon$-good approximate optimum path* (a path whose cost is no more than $(1+\epsilon)$ times that of an optimum path) for the weighted region optimal path problem.

In Section 1, we have defined and motivated the flow path problem, and stated our results. In Section 2, we provide some preliminary results on the geometry of optimum paths for flow path problems. In Section 3 we give a simple example where the $2\mathcal{D}$ version of the flow path problem consists of an exponential number of distinct segments through flow regions. In that section, we also provide a proof that the $3\mathcal{D}$ version of the flow path problem is PSPACE hard, which is the first known hardness result for the computational complexity of this problem. In Section 4, we provide the first known decision algorithm for the $2\mathcal{D}$ flow path problem. This decision algorithm is of theoretical interest only, but is proved by a rather interesting and unique inductive argument that repeatedly makes use of root separation bounds derived from the theory of real closed fields. In Section 5, we provide the first known approximation algorithm for the $2\mathcal{D}$ flow path problem, which is efficient for any given bounded error. In Section 6, we conclude the paper with some open problems.

## 2 Preliminaries

We first state some relevant properties of optimum paths for path planning problems within regions of translational flows:

**Proposition 1** *An optimum path is a simple path: it does not self-intersect.*

**Proposition 2** *For any two points $u$ and $u'$ in a flow region $r$, the face-wise optimum path connecting $u$ and $u'$ is a straight-line path with a control velocity of fixed direction and fixed maximum modulus $b_r$, as shown in Figure 2.*

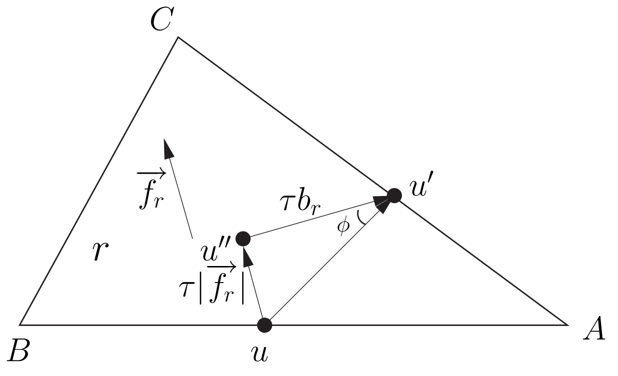


Figure 2: Face-wise optimum path is straight-line segment

The *face-wise optimum path* from $u$ to $u'$ is the path that takes the robot the minimum time among all paths that lie entirely inside $r$. It directly follows Proposition 2 that each optimum path in regions with flows is piecewise linear.

We now state the following two lemmas (refer to Figure 2) for $2\mathcal{D}$ flow path problems. We will apply them to develop approximation and decision algorithms.

**Lemma 1** *Let $r = \triangle ABC$ be a region with flow $\overrightarrow{f_r}$ and let $\beta = \angle BAC$, as shown in Figure 3. Let $u$ be a point on $\overline{AB}$ with distance $d$ to $A$ and $u'$ be a point on $\overline{AC}$ with distance $d'$ to $A$. Let $\alpha$ be the angle between $\overline{BA}$ and $\overrightarrow{f_r}$ and let $\theta$ be the angle between $\overline{uu'}$ and $\overline{BA}$. Then the face-wise optimum path from $u$ to $u'$ can be achieved by the point robot adopting a velocity with maximum magnitude and an angle of $\Phi = \arcsin(\frac{\sin(\alpha-\theta)}{\rho_r})$ from $\overline{uu'}$. Further, the cost $\tau$ of this path is $\frac{l^2}{b_r(\sqrt{l^2-T_1^2}+T_2)}$, where $l = |\overline{uu'}| = \sqrt{d^2+d'^2-2dd'\cos\beta}$, $T_1 = (d\sin\alpha - d'\sin(\alpha+\beta))/\rho_r$ and $T_2 = (d\cos\alpha - d'\cos(\alpha+\beta))/\rho_r$.*

**Proof** By Proposition 2, to go from $u$ to $u'$ within region $r$ with the minimum time, the robot need to take a control velocity $\overrightarrow{v_r}$ with a magnitude of $b_r$. Referring to Figure 2, we can draw a "virtual triangle" $\triangle uu'u''$ such that: $\angle u''uu' = \alpha - \theta$, $\overline{uu''} = \tau(u,u') \cdot |\overrightarrow{f_r}|$ and $\overline{u''u'} = \tau(u,u') \cdot b_r$. That is,

- $\overline{uu''}$ is the vector robot travels being carried by the flow;
- $\overline{u''u'}$ is the vector robot travels by its own control velocity $\overrightarrow{v_r}$ within region $r$;
- $\overline{uu'}$ is the composite vector of its movement.

By applying "Law of Sines" on this triangle, we have

$$\frac{\sin(\angle u''u'u)}{\sin(\angle u''uu')} = \frac{|\overline{uu''}|}{|\overline{u''u'}|} = \frac{|\overrightarrow{f_r}|}{b_r} = \frac{1}{\rho_r} \tag{1}$$

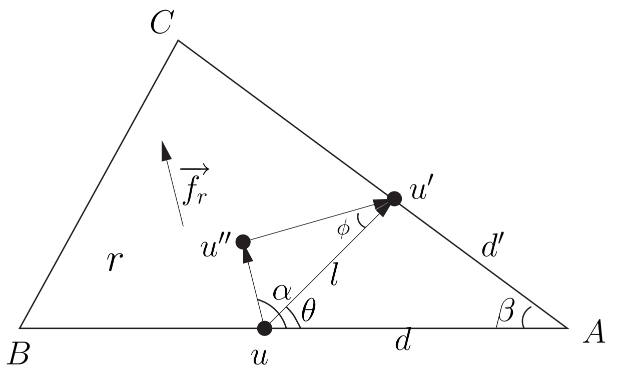


Figure 3: Computing $\tau(u, u')$

Note that there are two possible values for $\angle u''u'u$. However, since the smaller $\angle u''u'u$ is, the smaller $\tau(u, u')$ is, we will take $\phi = \angle u''u'u = \arcsin(\frac{\sin(\alpha-\theta)}{\rho_r})$.

To determine $\tau(u, u')$, we first need to evaluate $l = |\overline{uu'}|$. By applying "Law of Cosines" on triangle $\triangle u'uA$ we have

$$|\overline{uu'}|^2 = |\overline{uA}|^2 + |\overline{u'A}|^2 - 2\cos(\angle uAu') \cdot |\overline{uA}| \cdot |\overline{u'A}|,$$

and hence

$$l = \sqrt{d^2 + d'^2 - 2dd'\cos\beta}. \tag{2}$$

Here $\beta$ is a constant but $d$ and $d'$ change as $u$ and $u'$ change.

To determine $\theta$, we apply "Law of Sines" on this triangle and get

$$\frac{\sin(\angle u'uA)}{|\overline{u'A}|} = \frac{\sin(\angle u'Au)}{|\overline{uu'}|}.$$

Therefore, we have

$$\sin\theta = \frac{d' \cdot \sin\beta}{l} \tag{3}$$

and

$$\cos\theta = \frac{d - d'\cos\beta}{l}. \tag{4}$$

For $\triangle uu'u''$, we can use "Law of Sines" to get the following equations:

$$\frac{\sin(\angle u''uu')}{|\overline{u''u'}|} = \frac{\sin(\angle u''u'u)}{|\overline{uu''}|} \tag{5}$$

$$\frac{\sin(\angle u''uu')}{|\overline{u''u'}|} = \frac{\sin(\angle uu''u')}{|\overline{uu'}|} \tag{6}$$

Therefore, we have

$$\sin\phi = \sin(\angle u''u'u) = \frac{\sin(\alpha - \theta)}{\rho_r} \tag{7}$$

and

$$\cos\phi = \cos(\angle u''u'u) = \sqrt{1 - \frac{\sin^2(\alpha - \theta)}{\rho_r^2}}. \tag{8}$$

By applying "Law of Sines" on $\triangle u''uu'$ one more time, we have and thus

$$\begin{aligned}
\tau(u,u') &= \frac{|\overline{u''u'}|}{b_r} \\
&= \frac{\sin(\alpha-\theta)\cdot l}{\sin(\angle uu''u')\cdot b_r} \\
&= \frac{\sin(\alpha-\theta)\cdot l}{\sin(\pi-(\alpha-\theta+\phi))\cdot b_r} \\
&= \frac{\sin(\alpha-\theta)\cdot l}{b_r(\sin(\alpha-\theta)\cos\phi+\sin\phi\cos(\alpha-\theta))} \\
&= \frac{l}{b_r(\cos\phi+\sin\phi\cos(\alpha-\theta)/\sin(\alpha-\theta))} \\
&= \frac{l}{b_r(\cos\phi+\cos(\alpha-\theta)/\rho_r)} \\
&= \frac{l^2}{b_r(l\cos\phi+l\cos(\alpha-\theta)/\rho_r)}.
\end{aligned}$$

To represent $\tau(u,u')$ by constants and variables $d$ and $d'$, we first need to represent $\cos(\alpha-\theta)$ and $\sin(\alpha-\theta)$ by constants and variables $d, d'$. This can be done by expanding $\cos(\alpha-\theta)$ and $\sin(\alpha-\theta)$ and then replacing terms of $\sin\theta$ and $\cos\theta$ by right-hand side of Equation 3 and 4 respectively. Now we have

$$\sin(\alpha-\theta)=\frac{d\sin\alpha-d'\sin(\alpha+\beta)}{l} \tag{9}$$

and

$$\cos(\alpha-\theta)=\frac{d\cos\alpha-d'\cos(\alpha+\beta)}{l}. \tag{10}$$

Let $T_1=(d\sin\alpha-d'\sin(\alpha+\beta))/\rho_r$ and $T_2=(d\cos\alpha-d'\cos(\alpha+\beta))/\rho_r$. By applying Equation 8, 9 and 10 we have

$$\tau(u,u')=\frac{l^2}{b_r(\sqrt{l^2-T_1^2}+T_2)}. \tag{11}$$

This finishes the proof. ∎

## 3 Lower Bound Results

In this section we provide some lower bound results for flow path problems.

### 3.1 Exponential Size of $2\mathcal{D}$ Optimum Path

First we give a simple construction of an instance of $2\mathcal{D}$ flow path problem where the optimum path contains an exponential number of distinct straight-line segments.

**Theorem 1** *In the $2\mathcal{D}$ flow path problem with $O(1)$ regions and with flow rates specified by $N$ bits, the optimum path can consist of at least $\Omega(2^N)$ distinct straight-line segments through flow regions.*

**Proof** Refer to Figure 4. We construct a flow path problem where there are four unit square flow regions on the plane, each bordering the origin, and with unit magnitude flows that force a point robot with control velocity magnitude $2^{-N}$ starting at the source point $s$, which is very close to the origin, to spiral an exponential number of times around the origin before reaching the destination point $(1,0)$.

In particular, for $i = 0,1,2,3$, define angle $\theta_i = i\pi/2 + \pi/4$ and let the $i$th flow region be a unit square centered at point $(\frac{\sqrt{2}}{2}\cos(\theta_i), \frac{\sqrt{2}}{2}\sin(\theta_i))$, with unit flow direction $\theta_i + \pi/2$. Thus the concatenation of the flows of the four squares run in counterclockwise fashion around the origin. Starting at the origin, the goal point $(1,0)$ can only be reached by an exponential number of cycles by the point robot around the origin, where on each cycle the point can get an additional distance $c2^{-N}$ further from the origin, for a fixed constant $c > 0$. ∎

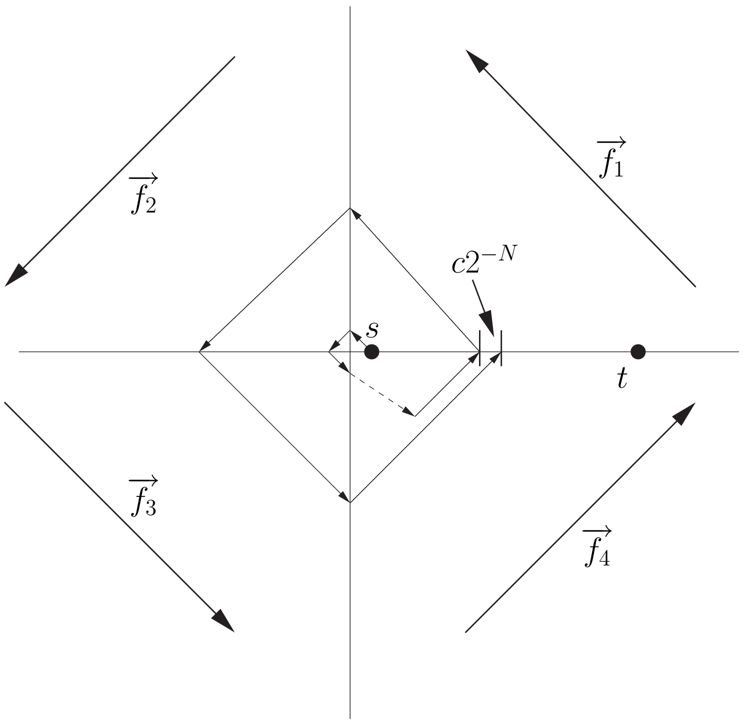


Figure 4: Optimum path may have $\Omega(2^N)$ segments

## 3.2 A PSPACE Hardness Result for $3\mathcal{D}$ Flow Path Problem

Next we prove that the $3\mathcal{D}$ flow path problem is PSPACE hard with respect to log-space reduction. We need to show that, for a polynomial space Turing Machine $M$ with given binary input of length $n$, we can construct an instance of the $3\mathcal{D}$ flow path problem with polynomial number of bits such that $M$ accepts the input iff there is a path connecting the source point and the destination point with a time cost no more than $T$, for some $T > 0$.

For any input $\omega$ of a polynomial space Turing Machine $M$, we can construct an input $\omega'$ of an equivalent linear space Turing Machine $M'$ by "padding" a polynomial number of empty characters to $\omega$. For example, if $M$ accepts an input only in $n^{c_1}$ space, we can construct $\omega'$ by adding $(n^{c_1} - n)$ empty characters to the end of $\omega$, such that $M'$ accepts $\omega'$ in linear space iff $M$ accepts $\omega$ in $n^{c_1}$ space. Therefore, we assume without loss of generality that $M$ accepts inputs in linear space. Further, we assume that $M$ accepts inputs only in $2^{cn}$ steps of computation, for some constant $c \geq 1$.

We use a construction given by Canny and Reif [5] in their proof of the NP hardness of the *3D* Euclidean shortest path problem. Given a Boolean formula with a list of $N$ variables $X$, they construct a system of a polynomial number of obstacles of polynomial size description to *simulate* this formula; for a distinguished source point $s$, a distinguished destination point $t$, as well as an intended path length $L$, there is a path of distance $\leq L$ between $s$ and $t$ avoiding these obstacles iff the formula is satisfiable.

For our proof, we use a system of this type that simulates a Boolean formula encoding the transition function of $M$. A robot will start at a specified source point and go through this system multiple times, each time simulating a single step of computation by $M$. With the introduction of flows into this system, each time the robot will take less than $\frac{T}{2^{cn}}$ time, for some $T > 0$, to go through the system if it follows a *legal path*, that is, a path corresponding to a transition of $M$'s configuration by one step of computation. Therefore, as long as $M$ accepts the input in $2^{cn}$ steps of computation, the robot will be able to reach a specified destination point in $T$ time. On contrary, the robot will either take more than $T$ time or even get trapped in the system if it ever takes an *illegal* path.

### 3.2.1 Canny and Reif's NP Hardness proof for *3D* Euclidean shortest path problem

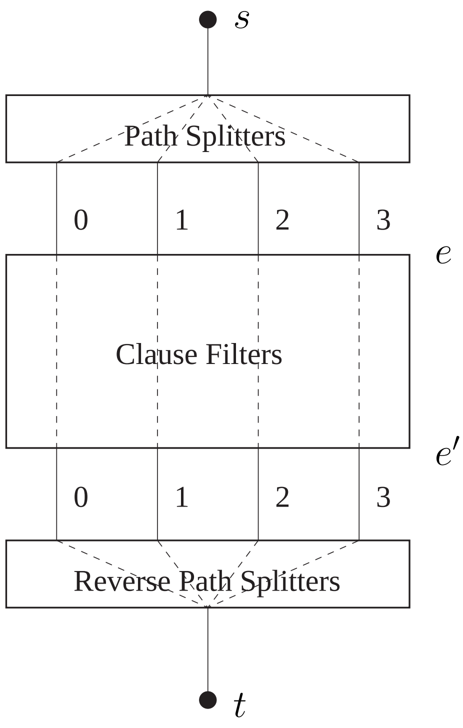


Figure 5: Canny and Reif's NP-hard construction

Canny and Reif's construction consists of three parts, as shown in Figure 5. The first part will generate by path splitting $2^N$ equal-length path classes, each of which encodes a possible variable assignment $A$ of $X$ and is denoted by $p_A$. For a distinguished obstacle segment $e$, each path class $p_A$ connects $s$ and a point $v_A$ on $e$. The second part will extend each path class $p_A$ to connect $v_A$ to a point $v'_A$ on another obstacle segment $e'$. The third part will merge the $2^N$ path classes to a single path, which eventually leads to $t$. Each path class $p_A$ will have length no more than $L$ if $A$

encodes a satisfying assignment. Otherwise, $p_A$ will be “stretched” by a significant amount so that its Euclidean length will be more than $L$.

The construction uses three types of substructures: i) *path splitter*, which doubles the number of path classes by splitting them; ii) *path shuffler*, which performs a perfect shuffle of path classes; and iii) *literal filter*, which filters for those paths that have a particular bit equal to zero or one in their encoding. Each substructure consists of a number of plates with $\epsilon$ width and spacing, each of which may contain one or two slits with $\epsilon$ width. Using path shufflers and literal filters, they construct *clause filters* each of which filters for those paths whose encodings satisfy a particular clause of the 3-SAT formula. (For details of the construction and usage of each substructure, please refer to [5].)

### 3.2.2 Adding flows to the substructures

Our proof technique differs from that of [5] in that, while they filter out path classes encoding non-satisfiable assignments by stretching them, we achieve so by forcing these path classes into a trap by overwhelming flow force so that they will never reach the destination point. At the same time, we also provide an overwhelming force along each path class encoding a satisfiable assignment so that the time cost of such path class will be extremely small.

Accordingly, we need to modify the substructures used in [5] by defining flows and control velocities. For the free space inside each slit, which is a rectangular volume with height and width of $\epsilon$ and length of $l_{slit}$, we define the modulus of control velocity of the point robot to be unit. The modulus of the control velocity is defined to be 0 everywhere else, either within the free space contained within the set of obstacles (not including slits), or inside the obstacles.

For each path splitter, we add strong flows each with modulus of $f$ on top of the first plate, between the first and second plates, and between the second and third plates, as shown in Figure 6. If four path classes (denoted by $p_1$, $p_2$, $p_3$, and $p_4$) come into a path splitter through Slit $S_{in}$ of Plate 1, they can choose to go left or right (without loss of optimality) in the space between Plate 1 and Plate 2, and cross Plate 2 through either Slit $S_1$ or Slit $S_2$. Either way, the path classes will move in the same direction in the space between Plate 2 and Plate 3, and eventually cross Plate 3 through Slit $S_{out}$. As a result, we will have eight path classes ($p'_1$, $p'_2$, $p'_3$, $p'_4$, $p''_1$, $p''_2$, $p''_3$, and $p''_4$), who have sub-paths of the same length inside this path splitter. If $l_{splitter}$ is the Euclidean length of each path class inside the path splitter, it will take $l_{splitter}/f + O(\epsilon)$ time to pass through the path splitter. (Recall that the point robot needs to pass through $O(1)$ slits of $\epsilon$ size with a unit control velocity.)

Similarly, we can add strong flows with the same modulus $f$ to each path shuffler so that it also has a similar property.

For each literal filter, we will add a strong flow with modulus $f$ so that every path whose encoding has a zero (one) in a specified bit will pass through the literal filter in $l_{filter}/f + O(\epsilon)$ time, as shown in Figure 7. Here $l_{filter}$ is the Euclidean distance traveled by each such path. We will also add a “bucket” to trap all the paths whose encodings have a one (zero, respectively) in the specified bit. If the robot comes from any of these paths, it will be pushed by the strong flow into the bucket. Since the robot has no control velocity in this region, it will never be able to leave the literal filter.

### 3.2.3 The proof

Now we are ready to prove the following theorem:

**Theorem 2** *The 3D flow path problem is PSPACE hard with respect to log-space reduction.*

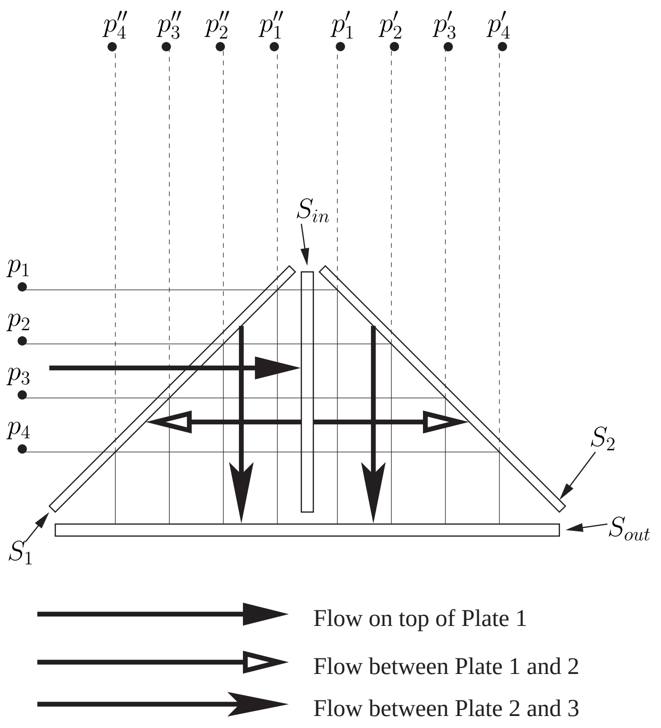


Figure 6: Path splitter with flows

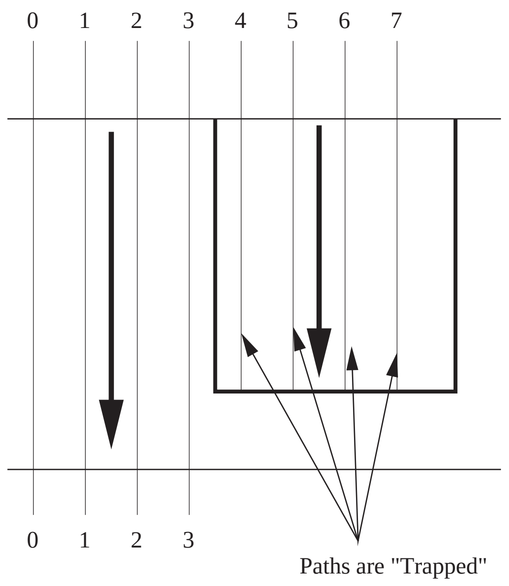


Figure 7: Path filter to trap paths having $b_{n-1} = 1$

We define a Boolean formula NEXT$(X, X')$ of $2N$ variables, where $N = cn + c'$ for some $c' = O(1)$. Here $X$ ($X'$, respectively) is a list of $N$ Boolean variables such that each assignment $A$ ($A'$, respectively) of $X$ ($X'$, respectively) encodes a configuration $\mathcal{C}(A)$ ($\mathcal{C}(A')$, respectively) of $M$. For assignment $(A, A')$ of $(X, X')$, NEXT$(A, A')$ is true iff $\mathcal{C}(A)$ and $\mathcal{C}(A')$ are valid configurations of $M$ and $\mathcal{C}(A')$ is reachable from $\mathcal{C}(A)$ by one step of computation by $M$. Let $m$ be the number of clauses of NEXT$(X, X')$ in the 3-SAT formula.

We apply the second part of Canny and Reif [5] construction to form a set of $3\mathcal{D}$ obstacles of polynomial size $f(n, m)$ with two distinguished obstacle segments $e, e'$. Note that instead we will use our "flowed" version of substructures described above. For any two boolean assignments $A$ and $A'$, there is a point $v_{A,A'}$ on edge $e$ and another point $v'_{A,A'}$ on $e'$. The Canny and Reif construction [5] ensures that point $v'_{A,A'}$ of segment $e'$ is reachable from point $v_{A,A'}$ of segment $e$ in (Euclidean) distance at most $L'$ if NEXT$(A, A')$ is true (and hence the configuration of $M$ may change from $\mathcal{C}(A)$ to $\mathcal{C}(A')$ in one step of computation).

With the addition of flows, a path that encodes a satisfying assignment will go from the point $v_{A,A'}$ of segment $e$ to $v'_{A,A'}$ of segment $e'$ in $O(L'/f + c'' \cdot \epsilon)$ time. Here $c'' = O(f(n, m))$ is the number of slits a path will pass through. Also, each path that does not encode a satisfying assignment will be trapped in one of the literal filters will never reach $v'_{A,A'}$.

Next we construct an obstacle segment $e_1$ that contains $2^N$ points, each of which encodes an assignment $A$ of $X$ and is denoted by $v_A$. We insert a number of path splitters between $e_1$ and $e$ so that each point $v_A$ on $e_1$ is connected with $v_{A,A'}$ on $e$ by a path $p'_{A,A'}$ for each assignment $A'$ of $X'$. Similarly, we provide for an obstacle segment $e'_1$ such that each point $v_{A'}$ on $e'_1$ is connected with $v_{A,A'}$ on $e$ by a path $p''_{A,A'}$.

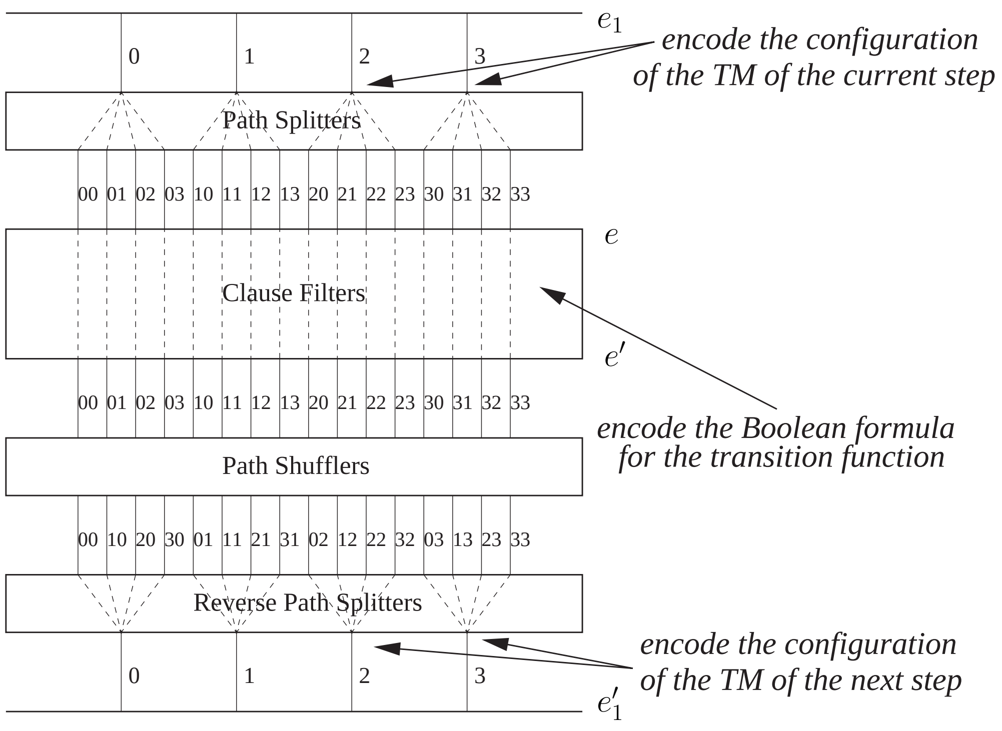


Figure 8: PSPACE construction

Finally, we provide for a translational "ribbon" that connects points of $e'_1$ back again to corresponding points of $e_1$. The ribbon consists of a sequence of connected rectangular flow strips each with a flow of modulus $f$ along the ribbon. Then the time of the point robot of move via this ribbon from a point $v'_{A'}$ of $e'_1$ to $v_{A'}$ of $e_1$ is at most $l_{ribbon}/f$, where $l_{ribbon}$ is the Euclidean length of the ribbon. We also define the modulus of the control velocity to be 0 inside the ribbon so that the point robot will not be allowed to move between points on $e_1$ and $e'_1$ that represent different configurations.

Therefore, during each cycle, if the robot starts at point $v_A$ on edge $e_1$, at the end it will reach point $v_{A'}$ on the same edge iff there is one step of computation that will bring $M$ from configuration $A$ to configuration $A'$. The purpose of this ribbon is to allow each of the single step transitions of the Turing machine to be repeated without changing the encoding of configurations.

Furthermore, we define the source point of the point robot to be a point $v_{A_0}$ of $e_1$ such that $A_0$ encodes the initial configuration of machine $M$, and we define the destination point of the point robot to be a point $v'_{A_f}$ of $e'_1$ such that $A_f$ encodes the accepting configuration of machine $M$.

We first assume that $M$ has an accepting computation. By our assumption, $M$ accepts the input in $2^{cn}$ steps. Therefore, it follows that there exists a path of time duration no more than $T = 2^{cn} \cdot ((L' + l_{ribbon})/f + c'' \cdot \epsilon)$ for the point robot to move between the source and destination points. If we choose $\epsilon \leq l_{slit}/(c'' \cdot 2^{2cn+c'+2})$ and $f \geq 2^{2cn+c'+2} \cdot (l_{ribbon} + L')/l_{slit}$, we have $T \leq l_{slit}/2^{cn+c'+1}$.

Next we examine the case when $M$ does not have an accepting computation of $2^{cn}$ steps. The only way a point robot can avoid being "trapped" inside one of the literal filters is to "cheat" by moving from one path class to another. The point robot can achieve such switching only inside one of the slits since the modulus of the control velocity is 0 everywhere else. As there are $N = cn + c'$ boolean variables, the number of path classes is $2^{cn+c'}$ and therefore the spacing between path classes in each slit is $l_{slit}/2^{cn+c'}$. Moving with a unit control velocity, it will at least take the point robot $l_{slit}/2^{cn+c'}$ time to switch from one path class to another, and therefore the total time duration of such a path is more than $T' = l_{slit}/2^{cn+c'}$.

It concludes that $M$ has an accepting computation using polynomial space iff the flow path problem has a path of time duration $T = l_{slit}/2^{cn+c'+1}$.

We still have to show that the transformation from an instance of the computation of $M$ to an instance of the $3\mathcal{D}$ flow path problem can be computed in log space; that is, there exists another deterministic Turing Machine $M'$ such that, given the description of $M$ (including its tape alphabet and transition function) along with an input $\omega$, $M'$ can construct an instance of the $3\mathcal{D}$ flow path problem using an auxiliary tape of $O(\log n)$ size. The construction described above uses a polynomial number of obstacles, each defined with polynomial number of bits. Further, the flow in each region can be defined with polynomial number of bits, as well as the positions of the source and destination points. Hence, the instance of flow path problem we use to simulate $M$ can be specified by polynomial number of bits. A Turing Machine $M'$ could "write" the specifications of these obstacles one by one, each time producing a single number of polynomial size. Since arithmetic computations of numbers of polynomial size can be carried out in log space, $M'$ will just need an auxiliary tape of log size in addition to the input and output tapes.

This finishes the proof of Theorem 2.

## 4 A Decision Algorithm for the $2\mathcal{D}$ Flow Path Problem

Here we develop a decision algorithm for the $2\mathcal{D}$ flow path problem of size $O(n)$ with coefficients each of which has $O(n)$ bits. We can assume, without loss of generality, that all flow regions are triangles.

Lemma 1 implies the following lemma:

**Lemma 2** *Let $s_r(d, d')$ be an optimum path segment for the robot within a convex flow region $r$ that begins at a distance $d$ along a boundary edge $e_1$ of $r$, and ends at a distance $d'$ along a boundary edge $e_2$ of $r$. Then there is a formula $F(t, x, x')$ of the existential theory of real closed fields (with free variables $t, x, x'$ and involving only a constant number of other variables; furthermore with a constant number of terms and with constant rational coefficients of at most $O(n)$ size), such that $F(\tau, d, d')$ is true iff $s_r(d, d')$ has time duration $\tau$.*

The *existential theory of real closed fields* is the logical system consisting of existentially quantified formulas, whose variables range over the real numbers, and whose formulas are constructed of inequalities of rational forms (these rational forms are arithmetic expressions involving these real variables and fixed rational constants which may be added and multiplied together) and the usual Boolean logical connectives AND, OR, NOT. Collins [6] gave a decision procedure for the existential theory of real closed fields that was improved by Canny [4] to run in polynomial space:

**Lemma 3** *Given a formula of the existential theory of real closed fields of length $n$, the formula can be decided in $n^{O(1)}$ space and $2^{O(n)}$ time, and the existentially quantified variables can be determined, up to exponential bit precision, within this computational complexity.*

Collins [6] proved a useful Lemma as a byproduct of his decision procedure:

**Lemma 4** *If the solution of a formula of length $n$ in the existential theory of real closed fields is not the zero vector $\mathbf{0}$, then it is of modulus at least $2^{-2^{cn}}$, for some constant $c > 0$.*

Now consider an optimum path $S(d, d')$ for the robot that begins at a distance $d$ along a boundary edge $e$ of a flow region $r$, and ends at a distance $d'$ along the same boundary edge $e$ of $r$, and does not visit any points of $e$ between these, but may pass through $f$ (where $f$ counts the repetitions) other flow regions. Then $S(d, d')$ consists of at most $O(f)$ straight-line segments, through flow regions. By Lemma 2, we have:

**Lemma 5** *There is a formula $F'(t, x, x')$ of the existential theory of real closed fields (with free variables $t, x, x'$ and involving only $O(f)$ other variables, and with $O(n^2)$ terms, and with constant rational coefficients of at most $O(n)$ size), such that $F'(\tau, d, d')$ is true iff $S(d, d')$ has time duration $\tau$.*

We define a hierarchy of paths that have a recursive structural characterization. We will use the term *structural complexity*, which should not be confused with the usual notion of computational complexity. Intuitively, structural complexity provides bound on the structure of the way paths crosses regions. Let a path have *structural complexity* 0 if it visits no flow region boundary edge more than once, except possibly at the start and final points of the path. Let a path $p$ have *structural complexity* $k$ if it does not have structural complexity $< k$ and $p = q_0 p_1 q_1 \ldots, p_h q_h$ where each $p_i$ is a path that passes through only one flow region, and each $q_i$ is a path of structural complexity $< k$ that begins and ends at the same flow region boundary edge. The following can be proved by induction on the number of distinct flow edges:

**Proposition 3** *The maximum structural complexity number of an optimum path in any 2$\mathcal{D}$ flow path problem with $n$ triangular flow regions is at most $n^{O(1)}$.*

We will derive, for any optimum path of structural complexity 0, a finite bound on the number of straight-line segments through flow regions. Consider an optimum path $S(d, d')$ of structural complexity 0 that begins and ends at a distance $d, d'$ respectively along a flow region boundary edge $e$ of a flow region $r$. Applying Lemma 4 to Lemmas 2 and 5, we have: either $d = d'$ or $|d' - d| \geq 2^{-2^{cn}}$, for some constant $c > 0$. This implies:

**Lemma 6** *Any optimum path of structural complexity* 0 *between two points consists of at most* $2^{2^{O(n)}}$ *straight-line segments through flow regions.*

In the following, let $E_0(n) = n$ and for $k > 0$, let $E_k(n) = 2^{E_{k-1}(n)}$.

Now suppose as an inductive assumption that, for some $k \geq 0$, any optimum path of structural complexity $k' < k$ between two points consists of at most $E_{2k'}(O(n))$ straight-line segments through flow regions. Applying Lemma 5, we can construct an existentially quantified formula of the theory of real closed fields with $E_{2k'}(O(n))$ variables, which is true iff there is an optimum path of structural complexity $k'$ and of length $\tau$ between the initial and final points.

Consider an optimum path $S(d, d')$ of structural complexity $k$ that begins and ends at a distance $d, d'$ respectively along a flow region boundary edge $e$ of a flow region $r$. Recall that since $S(d, d')$ has structural complexity $k$, it can be decomposed as $q_0 p_1 q_1 \ldots, p_h q_h$ where each $p_i$ is a path that passes through only one flow region, and each $q_i$ is a path of structural complexity $< k$ that begins and ends at the same flow region boundary edge. Applying again Lemma 4 to Lemmas 2 and 5, we now have: either $d = d'$ or $|d' - d| \geq 2^{-2^{O(E_{2(k-1)}(O(n)))}} \geq 1/E_{2k}(O(n))$. Hence we have that:

**Lemma 7** *Any optimum path of structural complexity $k$ between two points consists of at most* $E_{2k}(O(n))$ *straight-line segments through flow regions.*

Then applying the Canny [4] decision procedure (Lemma 3), we have that there is an algorithm that decides, within $E_{2k}(O(n))$ space and $2^{O(E_{2k}(O(n)))}$time, the $2\mathcal{D}$ flow path problem for optimum paths of structural complexity $k$. By Proposition 3 (which bounds the structural complexity number of an optimum path to be $n^{O(1)}$), we have one of our main results:

**Theorem 3** *There is an algorithm, with* $E_{2k}(O(n))$ *space and* $2^{O(E_{2k}(O(n)))}$ *time cost, for the* $2\mathcal{D}$ *flow path decision problem of size $n$, where $k \leq n^{O(1)}$ is the minimum structural complexity of an optimum path.*

## 5 An Efficient $\epsilon$-Approximation Algorithm

### 5.1 Approximating Optimum Paths by Discretization

The significance of the above algorithm is that the problem is decidable, but its complexity is far too high for practical implementation. A natural strategy to approximately solve the $2\mathcal{D}$ flow path problem is to discretize the polygonal decomposition of the $2\mathcal{D}$ space by inserting Steiner points on boundary edges. A directed discrete graph $\mathcal{G}$ is then constructed by interconnecting each pair of Steiner points or vertices on the boundary of the same region. Each directed edge $(u_1, u_2)$ connecting Steiner point (or vertex) $u_1$ to $u_2$ is assigned a weight that is equal to $\tau(u_1, u_2)$. A discrete search algorithm can be used to find a shortest path from the given source point $s$ to the destination point $t$ in $\mathcal{G}$. This shortest path can be used to approximate a "true" optimum path in the original continuous space.

The basic discretization scheme is to place $m$ Steiner points uniformly on each boundary edge, for some positive integer $m$. In $\mathcal{G}$ there will be $O(nm)$ points and $O(nm^2)$ edges and thus the time complexity of Dijkstra's algorithm is $O(nm^2 + nm\log(nm))$. Using the BUSHWHACK algorithm proposed by Sun and Reif (see ([22]), the optimum path in $\mathcal{G}$ can be computed in $O(nm \log(nm))$ time without visiting all edges. This scheme is easy to implement; however, it does not guarantee any relative (multiplicative) error bound of the approximation.

To compute an $\epsilon$-good approximate optimum path from $s$ to $t$ for a given error tolerance $\epsilon$, we can adopt a logarithmic discretization scheme similar to the one used by Aleksandrov *et al.* for the weighted region optimum path problem [1, 2]. The discretization scheme assumes that $\rho_r > 1$ for any region $r$.

We first introduce some notations. Throughout this section, we assume that $u$ is a point on a boundary edge of a region. We let $\tau_{min}(u)$ be the minimum cost of traveling (in either direction) along a straight-line path between $u$ and any point on an boundary edge not incident to $u$. For any boundary edge $e$, we let $u_e$ denote the point on boundary edge $e$ with maximum $\tau_{min}(u)$. We define $b_e$ to be the lesser of the maximum composite velocities of the robot traveling in either directions on $e$. For any vertex $v$, we use $R_v$ to give a lower-bound on the cost of traveling (in any possible path) between $v$ and any point on an boundary edge not incident to $v$.

For any boundary edge $e = \overline{v_1 v_2}$, $u_e$ divides edge $e$ into two segments $\overline{v_1 u_e}$ and $\overline{v_2 u_e}$, as shown in Figure 9. In the following we describe the placement of Steiner points $u_{i,1}, u_{i,2}, \cdots, u_{i,k}$ on each segment $\overline{v_i u_e}$ for $i = 1, 2$. The first Steiner point $u_{i,1}$ is placed on segment $\overline{v_i u_e}$ with distance $b_e R_{v_i} \epsilon$ to vertex $v_i$. The subsequent Steiner point $u_{i,j}$ is placed between $u_{i,j-1}$ and $u_e$ with a distance $\epsilon b_e \tau_{min}(u_{i,j-1})$ to $u_{i,j-1}$. We continue adding Steiner points until no more Steiner point can be added on $\overline{v_i u_e}$. That is, if $u_{i,j}$ is the last Steiner point added, no more Steiner point is inserted on segment $\overline{v_i u_e}$ if $|\overline{v_i u_{i,j}}| + \epsilon b_e \tau_{min}(u_{i,j}) \geq |\overline{v_i u_e}|$. Finally, we add $u_e$ as a Steiner point on $e$.

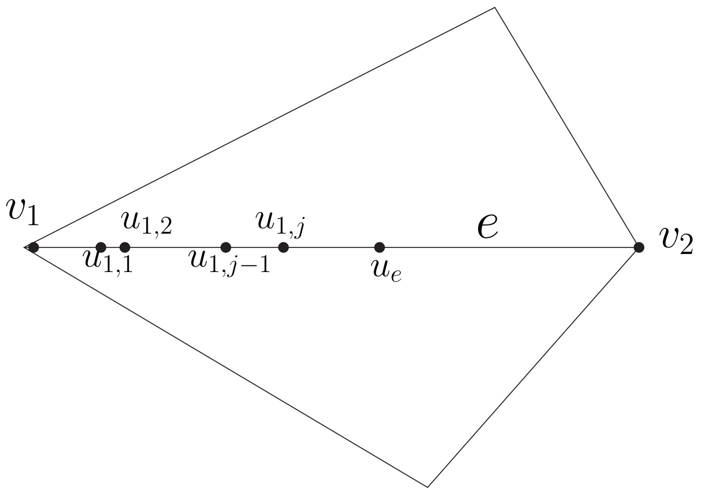


Figure 9: Adding Steiner points on boundary edge

Observe that, by this discretization scheme, instead of placing Steiner points with even spacing, we place Steiner points with a higher density in portions of $e$ closer to the endpoints than in portions closer to $u_e$, with the only exception that we do not place any Steiner point in the *vertex vicinities* $\overline{v_1 u_{1,1}}$ and $\overline{v_2 u_{2,1}}$. The intuition is that, roughly speaking, if an *optimum segment* (a segment of an optimum path) crosses $e$ at a point close to $u_e$, the segment will be relatively long. Therefore, it is always possible to find an *approximate segment* (a segment connecting two Steiner points) that "neighbors" the optimum segment; further, the cost of this approximate segment is no more than $(1 + \epsilon)$ times that of the cost of optimum segment. By substituting every optimum segment of an optimum path by an approximate segment, we will be able to construct a discrete path whose cost is no more than $(1+\epsilon)$ times that of the optimum path.

Once the Steiner points are chosen, we can again construct a weighted directed graph $\mathcal{G}_\epsilon$ and then apply either the Dijkstra's algorithm or the BUSHWHACK algorithm to compute a shortest path from $s$ to $t$ in $\mathcal{G}_\epsilon$, and use this path to approximate a true optimum path in the original polygonal decomposition.

In the rest of this section, we will give an upper bound on the size of the discretization generated, as well as prove that this discretization indeed contains $\epsilon$-good approximate optimum paths.

## 5.2 Bounding the Size of the Discretization

We first establish several lemmas on some properties of the parameters used for constructing the discretization.

Let $r$ be a triangular region with boundary edges $e, e'$, and $e''$, and let $u$ be an internal point on $e$, as shown in Figure 10. We denote $\inf\{\tau(u,q)|q \in e'\}$ and $\inf\{\tau(q,u)|q \in e'\}$ by $\tau_{min}(u,e')$ and $\tau_{min}(e',u)$, respectively. Let $u_{min}(u,e')$ ($u_{min}(e',u)$, respectively) be the point on $e'$ such that $\tau(u, u_{min}(u,e')) = \tau_{min}(u,e')$ ($\tau(u_{min}(e',u),u) = \tau_{min}(e',u)$, respectively). Further, we denote $\min\{\tau_{min}(u,e'), \tau_{min}(u,e'')\}$ and $\min\{\tau_{min}(e',u), \tau_{min}(e'',u)\}$ by $\tau_{min}(u,r)$ and $\tau_{min}(r,u)$, respectively, and define $u_{min}(u,r)$ and $u_{min}(r,u)$ accordingly. Therefore,

$$\tau_{min}(u) = \min\{\tau_{min}(u,r), \tau_{min}(r,u), \tau_{min}(u,r'), \tau_{min}(r',u)\},$$

where $r'$ is the other flow region incident to $e$.

If $u$ is a vertex of a region, we can define the above notations in an analogous manner. The difference is that in this case $u$ may have more than two incident regions, and for each region there is only one boundary edge "facing" $u$.

The next two lemmas show that $u_{min}(u,e'), \tau_{min}(u,e'), u_{min}(e',u)$, and $\tau_{min}(e',u)$ (and hence $\tau_{min}(u)$ and $u_e$) all can be determined in constant time.

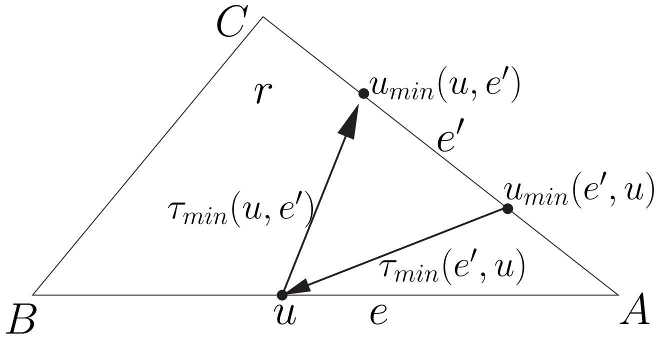
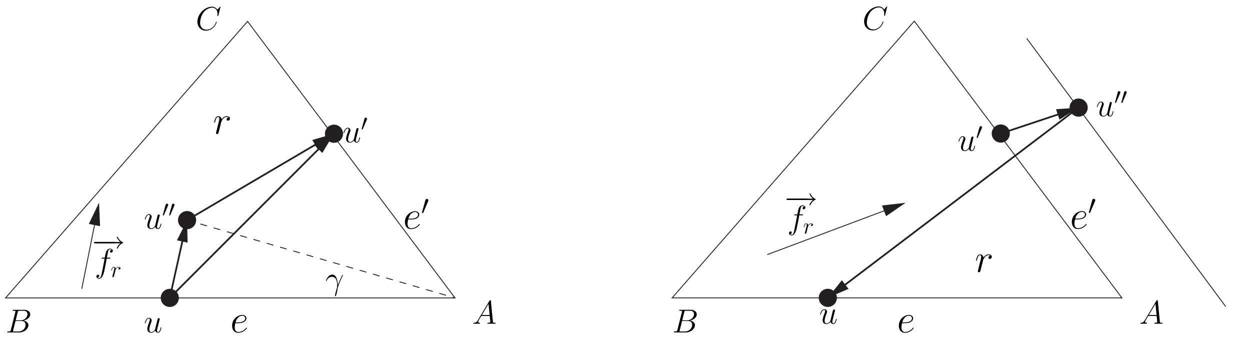


Figure 10: Points with minimum distance to/from $u$

**Lemma 8** *Let $d'' = (d \cdot (\sin\alpha + \rho_r \cdot \cos\beta))/(\sin(\alpha+\beta)+\rho_r)$, where $d, \alpha, \beta, \rho_r, b_r$ and $\overrightarrow{f_r}$ are defined as previously.*
**i)** if $0 \le d'' \le |\overline{AC}|$: *$u_{min}(u,\overline{AC})$ is the point with distance $d''$ from $A$ on $\overline{AC}$ and $\tau_{min}(u,\overline{AC}) = (d \cdot \sin\beta)/(|\overrightarrow{f_r}| \cdot (\sin(\alpha+\beta)+\rho_r))$;*
**ii)** if $d'' < 0$: *$u_{min}(u,\overline{AC}) = A$ and $\tau_{min}(u,\overline{AC}) = \tau(u,A)$;*
**iii)** if $d'' > |\overline{AC}|$: *$u_{min}(u,\overline{AC}) = C$ and $\tau_{min}(u,\overline{AC}) = \tau(u,C)$.*

**Proof** Refer to Figure 11.a. Let $u'$ be the point on line $\overline{AC}$ (not necessarily between $A$ and $C$) with property that $\angle u''u'A = \pi/2$. It is obvious that $u_{min}(u, \overline{AC}) = u'$ if $u'$ is between $A$ and $C$. Here $u''$ is defined the same way as in the proof of Lemma 1.



(a) Minimum cost from $u$ to a boundary edge (b) Minimum cost from a boundary edge to $u$

Figure 11: Computing $\tau(u, e')$ and $\tau(e', u)$

Let $\gamma = \angle u''Au$. Applying "Law of Sines" on $\triangle u''u'A$ and $\triangle u''uA$, we have the following equations:

$$\frac{\sin\alpha}{|\overline{Au''}|} = \frac{\sin\gamma}{|\overline{uu''}|}, \tag{12}$$

$$|\overline{u'u''}| = \sin(\beta - \gamma) \cdot |\overline{Au''}|. \tag{13}$$

Replacing $|\overline{u'u''}|$ by $\sin(\beta - \gamma) \cdot |\overline{Au''}|$ in Equation 12 and considering the fact that $\rho_r \cdot |\overline{uu''}| = |\overline{u''u'}|$,we get

$$\gamma = \arctan\left(\frac{\sin\beta}{\frac{\rho_r}{\sin\alpha} + \cos\beta}\right) \tag{14}$$

Therefore, we can determine $|\overline{Au'}|$, $|\overline{u'u''}|$ and $|\overline{uu''}|$ as the following:

$$|\overline{uu''}| = \frac{\sin\gamma \cdot |\overline{Au}|}{\sin(\alpha + \gamma)} = \frac{\sin\beta \cdot d}{\sin(\alpha + \beta) + \rho_r}, \tag{15}$$

$$|\overline{u'u''}| = \rho_r \cdot |\overline{uu''}| = \frac{\rho_r \cdot \sin\beta \cdot d}{\sin(\alpha + \beta) + \rho_r}, \tag{16}$$

$$|\overline{Au'}| = \frac{|\overline{u'u''}|}{\tan(\beta - \gamma)} = \frac{d \cdot (\sin\alpha + \rho_r \cdot \cos\beta)}{\sin(\alpha + \beta) + \rho_r}. \tag{17}$$

Here again $d$ is the distance from $u$ to $A$.

$|\overline{Au'}|$ represents the distance from $A$ to $u'$ along the direction of ray $\overline{AC}$. If $u'$ is on boundary edge $\overline{AC}$, i.e., $0 \leq |\overline{Au'}| \leq |\overline{AC}|$, $u_{min}(u, \overline{AC}) = u'$ and $\tau_{min}(u, \overline{AC}) = \frac{|\overline{uu''}|}{|\overrightarrow{f_r}|} = \frac{d \cdot \sin\beta}{|\overrightarrow{f_r}| \cdot (\sin(\alpha + \beta) + \rho_r)}$. If $|\overline{Au'}| < 0$, $u_{min}(u, \overline{AC}) = A$ and $\tau_{min}(u, \overline{AC}) = \tau(u, A)$; if $|\overline{Au'}| > |\overline{AC}|$, $u_{min}(u, \overline{AC}) = C$ and $\tau_{min}(u, \overline{AC}) = \tau(u, C)$. ∎

**Lemma 9** *Let $d'' = (d \cdot (\rho_r \cdot \cos\beta - \sin\alpha))/(\rho_r - \sin(\alpha+\beta))$.*
**i)** if $0 \le d'' \le |\overline{AC}|$: *$u_{min}(\overline{AC}, u)$ is the point with distance $d''$ from $A$ on $\overline{AC}$ and $\tau_{min}(\overline{AC}, u) = (d \cdot \sin\beta)/(|\overrightarrow{f_r}| \cdot (\rho_r - \sin(\alpha+\beta)))$;*
**ii)** if $d'' < 0$: *$u_{min}(\overline{AC}, u) = A$ and $\tau_{min}(\overline{AC}, u) = \tau(A, u)$;*
**iii)** if $d'' > |\overline{AC}|$: *$u_{min}(\overline{AC}, u) = C$ and $\tau_{min}(\overline{AC}, u) = \tau(C, u)$.*

**Proof** Observe that $u_{min}(\overline{AC}, u)$ is the point on line $\overline{AC}$ with the property that $\angle u''uA = \pi/2 - \angle CAB$ (as shown in Figure 11.b), if such a point lies between $A$ and $C$. Here $u''$ is the point such that $\rho_r|\overline{u_{min}(e', u)u''}| = |\overline{u''u}|$ and that $\overrightarrow{u_{min}(e', u)u''}$ has the same direction as $\overrightarrow{f_r}$. We can prove Lemma 9 with similar steps used by the proof for Lemma 8. ∎

With Lemmas 8 and 9, we can prove that $\tau_{min}$ as a function of $u$ has the following property:

**Lemma 10** *$\tau_{min}(u)/|\overline{Au}|$ is a non-increasing function when point $u$ moves from $A$ towards $B$ along boundary edge $\overline{AB}$.*

Since $\tau_{min}(u) = \min\{\tau_{min}(u, r),\ \tau_{min}(u, r'),\ \tau_{min}(r, u),\ \tau_{min}(r', u)\}$, to prove Lemma 10 it suffices to establish the following lemma:

**Lemma 11** *Let $r$ be a region incident to boundary edge $\overline{AB}$. $\tau_{min}(u, r)/|\overline{Au}|$ is a non-increasing function when $u$ moves from $A$ towards $B$ on $\overline{AB}$. Also, $\tau_{min}(r, u)/|\overline{Au}|$ is a non-increasing function when $u$ moves from $A$ towards $B$ on $\overline{AB}$.*

**Proof** Let $r = \triangle ABC$. For any $u \in \overline{AB}$, let $u_{min}(u, r)$ be the point on $\overline{BC}$ or $\overline{AC}$ such that $\tau(u, u_{min}(u, r)) = \tau_{min}(u, r)$.

We first claim that $u_{min}(u, r)$ can not be vertex $C$ for any $u$ on $\overline{AB}$. As shown in Figure 12.a, let $u''$ be the vertex of the "virtual triangle" defined as previously. Therefore we have $\rho_r|\overline{uu''}| = |\overline{Cu''}|$ and $\angle u''uA$ is equal to the angle between $\overrightarrow{f_r}$ and $\overline{uA}$, and $\angle u''uB$ is equal to the angle between $\overrightarrow{f_r}$ and $\overline{uB}$. It is easy to prove that either $\angle u''CA$ or $\angle u''CB$ is less than $\pi/2$, given the fact that $\rho_r > 1$. Suppose $\angle u''CA < \pi/2$. Then there is a point $u'_1$ on $\overline{AC}$ with distance $\delta$ to $C$ such that $\overline{u''u'_1} < \overline{u''C}$. Therefore, there exists a point $u''_1$ between $u$ and $u''$ on $\overline{uu''}$ such that $\rho_r|\overline{uu''_1}| = |\overline{u'_1u''_1}|$ and thus $\tau(u, u'_1) < \tau(u, C)$.

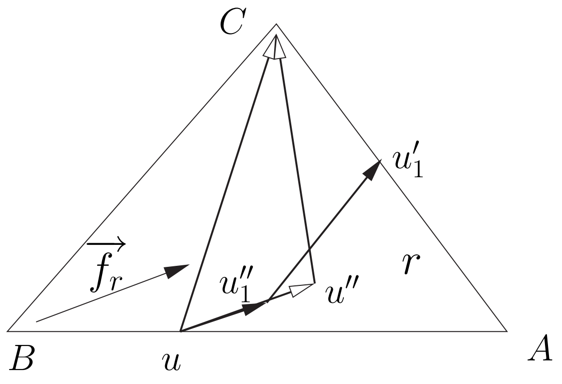


Figure 12: $u_{min}(u, r)$ cannot be vertex $C$

This shows that when $u$ is moving from $A$ to $B$ along $\overline{AB}$, $u_{min}(u, r)$ will be either $A$ or $B$, or some internal point on $\overline{AC}$ or $\overline{BC}$. If $u_{min}(u, r)$ is $B$ or some internal point on $\overline{BC}$, $\tau_{min}(u, r)/|\overline{Au}|$

is decreasing. If $u_{min}(u,r)$ is $A$, $\tau_{min}(u,r)/|\overline{Au}| = 1/b_{A,B}$ and thus is a constant. Similarly, if $u_{min}(u,r)$ is an internal point on $\overline{AC}$, $\tau_{min}(u,r)/|\overline{Au}|$ is a constant following Lemma 8. Therefore, in all cases, $\tau_{min}(u,r)/|\overline{Au}|$ is non-increasing.

With a similar argument we can prove that, for each $r$ incident to boundary edge $\overline{AB}$, $\tau_{min}(r,u)/|\overline{Au}|$ is a non-increasing function when $u$ moves from $A$ towards $B$ on $\overline{AB}$. This finishes the proof. ∎

Next, we show how to compute $b_e$ and $R_v$. As shown in Figure 13, $r_1$ and $r_2$ are the two regions incident to $e = \overline{v_1v_2}$, and $\alpha_1$ ($\alpha_2$, respectively) is the angles between $\overline{v_2v_1}$ and $\overrightarrow{f_{r_1}}$ ($\overrightarrow{f_{r_2}}$, respectively). Therefore, the maximum velocity $b_{v_1,v_2}$ of the robot traveling from $v_1$ to $v_2$ is the greater of the two maximum velocities specified by the two regions. That is, $b_{v_1,v_2} = \max\{\sqrt{b_{r_1}^2 - \sin^2\alpha_1 \cdot |\overrightarrow{f_{r_1}}|^2} + |\overrightarrow{f_{r_1}}| \cdot \cos\alpha_1, \sqrt{b_{r_2}^2 - \sin^2\alpha_2 \cdot |\overrightarrow{f_{r_2}}|^2} + |\overrightarrow{f_{r_2}}| \cdot \cos\alpha_2\}$. Similarly, we can define $b_{v_2,v_1}$ and let $b_e = \min\{b_{v_1,v_2}, b_{v_2,v_1}\}$. Note that $b_e$ is the minimum effective velocity of the robot traveling on boundary edge $e$ in any of the two directions.

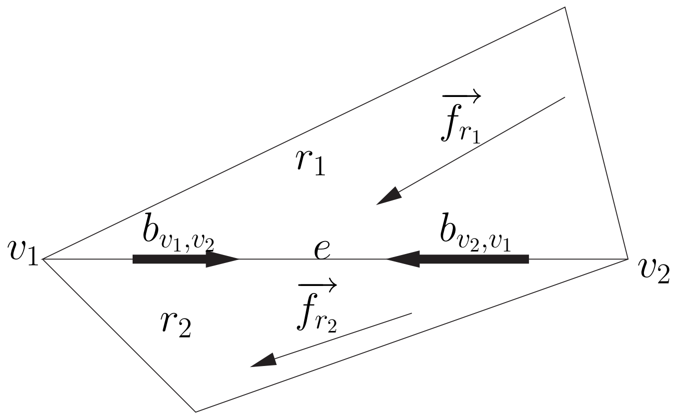


Figure 13: Effective velocity on boundary edge

For any vertex $v$ and any triangular region $r$ incident to $v$, we define $d_{min}(v,r)$ to be the minimum Euclidean distance from $v$ to the boundary edge of $r$ that is not incident to $v$. We define the *radius* of $v$ to be the following: $R_v = \min\{\frac{d_{min}(v,r)}{b_r + |\overrightarrow{f_r}|} \mid r \text{ is incident to } v\}$. Observe that while $\tau_{min}(v)$ is the minimum cost of a one-segment path between $v$ and any point on the boundary of the union of regions incident to $v$, $R_v$ provides a lower bound on the minimum cost of any path between $v$ and any point on the boundary of the union, as stated by the following lemma:

**Lemma 12** *For any vertex $v$ and any point $u$ on the boundary of the union of all regions incident to $v$, it takes at least $R_v$ time for a robot to traverse from $v$ to $u$, or from $u$ to $v$.*

Now we are ready to prove an upper bound on the number of Steiner points placed on each boundary edge $e$ with the following theorem:

**Theorem 4** *For a given $\epsilon$, the number of Steiner points added on $e$ according to the discretization scheme described above is $O(C'_e\frac{1}{\epsilon}(\log\frac{1}{\epsilon} + C''_e))$, where $C'_e$ and $C''_e$ are determined by the geometry as well as $b_r$ and $\overrightarrow{f_r}$ of each region $r$ incident to $e$.*

**Proof** Refer to Figure 9. For any $j \geq 1$ and $i = 1, 2$, we have the following inequalities:

$$\begin{aligned}
|\overline{v_i u_{i,j}}| &= |\overline{v_i u_{i,j-1}}| + |\overline{u_{i,j-1} u_{i,j}}| \\
&= |\overline{v_i u_{i,j-1}}| + \epsilon b_e \tau_{min}(u_{i,j-1}) \\
&\geq |\overline{v_i u_{i,j-1}}| + \epsilon b_e \frac{|\overline{v_i u_{i,j-1}}| \cdot \tau_{min}(u_e)}{|\overline{v_i u_e}|} \text{ (following Lemma 10)} \\
&\geq |\overline{v_i u_{i,j-1}}| + \epsilon b_e \frac{|\overline{v_i u_{i,j-1}}| \cdot \tau_{min}(u_e)}{|e|} \\
&= |\overline{v_i u_{i,j-1}}|(1 + \epsilon b_e \frac{\tau_{min}(u_e)}{|e|}) \\
&= b_e R_{v_i} \epsilon \cdot (1 + \epsilon b_e \frac{\tau_{max}(e)}{|e|})^{j-1}.
\end{aligned}$$

Let $C_e = b_e \frac{\tau_{max}(e)}{|e|}$. Suppose $k_i$ is the number of Steiner points inserted on segment $\overline{v_i u_e}$. As $|\overline{v_i u_{i,k_i}}| < |e|$, we have $b_e R_{v_i} \epsilon \cdot (1 + C_e \cdot \epsilon)^{k_i - 1} < |e|$ and thus $k_i < 1 + \frac{\log \frac{|e|}{b_e R_{v_i} \epsilon}}{\log(1 + C_e \cdot \epsilon)}$. Assuming that $\epsilon$ is small enough so that $\epsilon \cdot C_e \leq 0.5$, we have

$$k_i < 1 + \frac{5}{4C_e} \cdot \frac{1}{\epsilon} \cdot (\log \frac{1}{\epsilon} + \log \frac{|e|}{b_e R_{v_i}}).$$

The total number of Steiner points placed on boundary edge $e = \overline{v_1 v_2}$ is $k_1 + k_2 + 1 = O(C'_e \frac{1}{\epsilon}(\log \frac{1}{\epsilon} + C''_e))$, where $C'_e = \frac{1}{C_e}$ and $C''_e = \log \frac{|e|}{b_e \sqrt{R_{v_1} R_{v_2}}}$. ∎

To give upper bounds for both $C'_e$ and $C''_e$, we introduce three parameters $\theta_{min}$, $\rho_{min}$ and $\lambda$. We let $\theta_{min}$ be the smallest angle of any triangular region and let $\rho_{min}$ be the minimum $b_r$ among all regions. We define $\lambda$ as the following:

$$\lambda = \max\{b_r / b_{r'} | \text{regions } r \text{ and } r' \text{ are adjacent}\}.$$

$\lambda$ indicates how drastic the velocity bound of the robot can change from one region to another. In practice, since the triangular decomposition is usually a result of discretization, $\lambda$ is not very large in many cases. Further, we let $C_{skew} = \frac{\lambda \cdot (\rho_{min} + 1)}{\sin \theta_{min} \cdot (\rho_{min} - 1)}$ and call it the "skewness" parameter of the space.

We have the following lemma:

**Lemma 13** *$C'_e = O(C_{skew})$ and $C''_e = O(\log C_{skew})$.*

**Proof** We first show that $C'_e = O(C_{skew})$. Let $e = \overline{v_1 v_2}$ be a boundary edge with adjacent regions $r_1, r_2$. Let $e_{i,1}, e_{i,2}$ be the other two boundary edges of $r_i$. Let $v_{mid}$ be the mid point of $\overline{v_1 v_2}$. For each region $r_i$, the Euclidean distance from $v_{mid}$ to any point on $e_{i,1}$ and $e_{i,2}$ is lower-bounded by $\sin \theta_{min} |e|/2$. The cost of traveling from $v_{mid}$ to any point on $e_{i,1}$ and $e_{i,2}$ is thus lower-bounded by $\sin \theta_{min} |e| / (b_{r_i} + |\overrightarrow{f_{r_i}}|)$. Therefore, we have $\tau_{min}(v_{mid}) \geq \sin \theta_{min} |e| / \max\{(b_{r_i} + |\overrightarrow{f_{r_i}}|) \mid i = 1, 2\}$.

On the other hand, $b_e$, the minimum effective velocity of traveling on either direction on boundary edge $e$, is lower-bounded by $\min\{b_{r_1} - |\overrightarrow{f_{r_1}}|, b_{r_2} - |\overrightarrow{f_{r_2}}|\}$. We can have the following inequality:

$$
\begin{aligned}
C'_e &= \frac{1}{C_e} = \frac{|e|}{b_e \tau_{max}(e)} \\
&= \frac{|e|}{b_e \tau_{min}(u_e)} \le \frac{|e|}{b_e \tau_{min}(v_{mid})} \\
&\le \frac{|e|}{b_e \sin\theta_{min} |e| / (\max\{b_{r_1} + |\overrightarrow{f_{r_1}}|, b_{r_2} + |\overrightarrow{f_{r_2}}|\})} \\
&= \frac{\max\{b_{r_1} + |\overrightarrow{f_{r_1}}|, b_{r_2} + |\overrightarrow{f_{r_2}}|\}}{b_e \sin\theta_{min}} \\
&\le \frac{\max\{b_{r_1} + |\overrightarrow{f_{r_1}}|, b_{r_2} + |\overrightarrow{f_{r_2}}|\}}{\min\{b_{r_1} - |\overrightarrow{f_{r_1}}|, b_{r_2} - |\overrightarrow{f_{r_2}}|\} \sin\theta_{min}} \\
&\le \frac{(1 + 1/\rho_{min}) \max\{b_{r_1}, b_{r_2}\}}{(1 - 1/\rho_{min}) \min\{b_{r_1}, b_{r_2}\} \sin\theta_{min}} \\
&\le \frac{(1 + 1/\rho_{min})\lambda}{(1 - 1/\rho_{min}) \sin\theta_{min}} = \frac{(\rho_{min} + 1)\lambda}{(\rho_{min} - 1) \sin\theta_{min}} \\
&= C_{skew}.
\end{aligned}
$$

For $C''_e$, recall that for $i = 1, 2$, we have

$$R_{v_i} \ge |e| \sin^2\theta_{min} / \max\{|\overrightarrow{f_r}| + b_r \mid \text{region } r \text{ is incident to } v_i\}.$$

Therefore,

$$
\begin{aligned}
C''_e &= \log \frac{|e|}{b_e \sqrt{R_{v_1} R_{v_2}}} \\
&\le \log \frac{|e| \max\{|\overrightarrow{f_r}| + b_r \mid \text{region } r \text{ is incident to } v_1 \text{ or } v_2\}}{\min\{b_{r_1} - |\overrightarrow{f_{r_1}}|, b_{r_2} - |\overrightarrow{f_{r_2}}|\} |e| \sin^2\theta_{min}} \\
&\le \log \frac{\lambda(\rho_{min} + 1)}{(\rho_{min} - 1) \sin^2\theta_{min}} \\
&= O(\log C_{skew}).
\end{aligned}
$$

Note that none of $C'_e, C''_e$ and $C_{skew}$ is a function of $\epsilon$. ∎

We have the following theorem:

**Theorem 5** *For the nonuniform discretization scheme, the total number of Steiner points added into the triangular decomposition is* $O\left(\frac{C_{skew} \cdot n}{\epsilon} \log \frac{C_{skew}}{\epsilon}\right)$*, where $n$ is the number of triangular regions in the decomposition.*

The dependence of $C_{skew}$ on $\lambda$ implies that the variations of the velocity bound of the robot in different regions will affect the complexity of this approximation algorithm. If the velocity bound is uniform in all regions, less Steiner points are needed to guarantee an $\epsilon$-approximation. On the contrary, if the velocity bound changes drastically in adjacent regions, it will take much more Steiner points to achieve the same approximation bound.

## 5.3 Bounding the Error of the Approximation

The discrete graph $\mathcal{G}_\epsilon$ constructed from the discretization described above has $O(\frac{C_{skew} \cdot n}{\epsilon} \log \frac{C_{skew}}{\epsilon})$ points and $O(n(\frac{C_{skew}}{\epsilon} \log \frac{C_{skew}}{\epsilon})^2)$ edges. The time complexity of computing the shortest path in $\mathcal{G}$ is $O(\frac{C_{skew} \cdot n}{\epsilon}(\frac{C_{skew}}{\epsilon} \log \frac{C_{skew}}{\epsilon} + \log n) \log \frac{C_{skew}}{\epsilon})$ by Dijkstra's algorithm, or $O(\frac{C_{skew} \cdot n}{\epsilon}(\log \frac{C_{skew}}{\epsilon} + \log n) \log \frac{C_{skew}}{\epsilon})$ by the BUSHWHACK algorithm. In this subsection we analyze how well a discrete path can approximate an optimum path in the original continuous space.

As we have mentioned earlier, an optimum path $p_{opt}$ from a given source point $s$ to a given destination point $t$ in the original continuous space is piecewise linear. It only bends on the boundary of regions. We establish the following theorem that provides a bound on the error of using a discrete path to approximate an optimum path.

Let $p$ be a linear path with a cost of $d(p)$. In the following we first describe how to construct a discrete path $p'$ that "neighbors" a given linear path $p$, and then we show that the cost of such a path $p'$ is no more than $(1 + 9\epsilon)d(p)$ by using several lemmas.

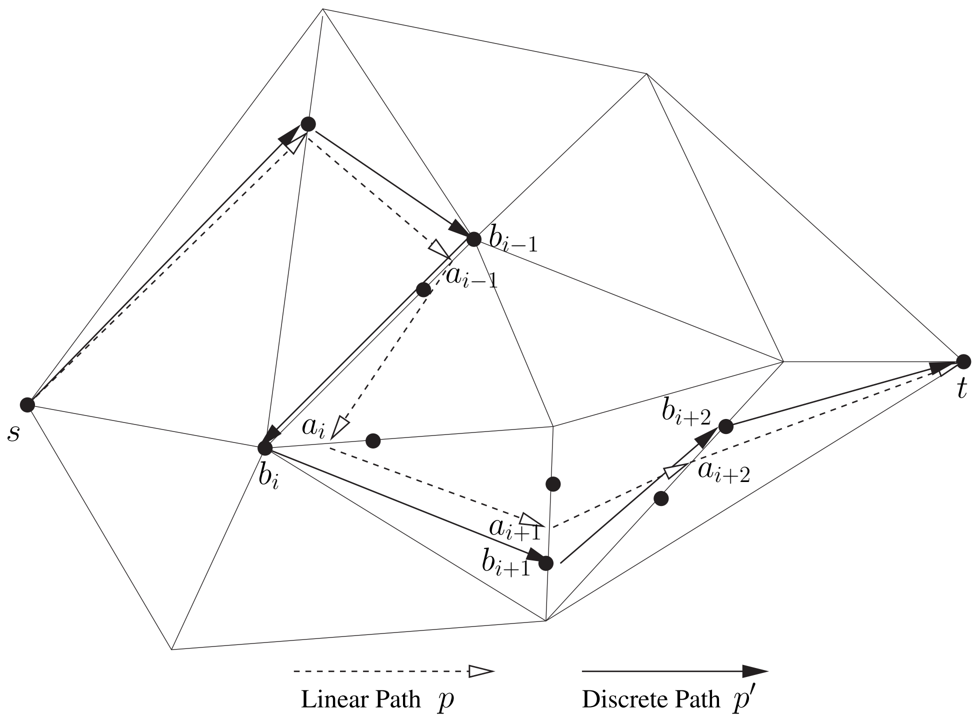


Figure 14: Discrete path that neighbors a linear path

Let $s = a_1, a_2, \cdots, a_{k-1}, a_k = t$ be the points where a linear path $p$ bends, as shown in Figure 14. We need to construct a discrete path with bending points $s = b_1, b_2, \cdots, b_{k-1}, b_k = t$ such that each $b_i$ is either a Steiner point or a vertex. As path $p$ only bends on the boundary of regions, each bending point $a_i$ is either between two Steiner points on the same boundary edge, or between a vertex and its neighboring Steiner point.

**1** Suppose that $a_i$ is between two Steiner points $u_j, u_{j+1}$ on boundary edge $e = \overline{v_{i_1} v_{i_2}}$. Without loss of generality, suppose that $u_j$ is between $u_{j+1}$ and $v_{i_1}$. Then we choose $b_i$ to be $u_j$ if $u_j$ is between $v_{i_1}$ and $u_e$; if otherwise, we choose $b_i$ to be $u_{j+1}$.

**2** Suppose that $a_i$ is between vertex $v_{i_1}$ and Steiner point $u_1$ on boundary edge $e = \overline{v_{i_1}v_{i_2}}$. Then we choose $b_i$ to be $v_{i_1}$.

To evaluate $d(p')$, we need to estimate the cost $\tau(b_j, b_{j+1})$ of each segment $\overline{b_j b_{j+1}}$. In particular, we need to compare $\tau(b_j, b_{j+1})$ against $\tau(a_j, a_{j+1})$ in various situations. In the following, we show:

**Lemma 14** *Assuming $\epsilon \leq \frac{1}{3}$, if both $b_j$ and $b_{j+1}$ are Steiner points, then $\tau(b_j, b_{j+1}) \leq (1 + 3\epsilon)\tau(a_j, a_{j+1})$. Also, if one of $b_j$ and $b_{j+1}$ is a vertex $v$ of the triangular decomposition, then $\tau(b_j, b_{j+1}) \leq (1 + \frac{3}{2}\epsilon)\tau(b_j, b_{j+1}) + \epsilon R_v$. Further, if both $b_j$ and $b_{j+1}$ are vertices of the triangular decomposition, then $\tau(b_j, b_{j+1}) \leq \tau(a_j, a_{j+1}) + \epsilon R_{v'} + \epsilon R_{v''}$, where $v' = b_j$ and $v'' = b_{j+1}$.*

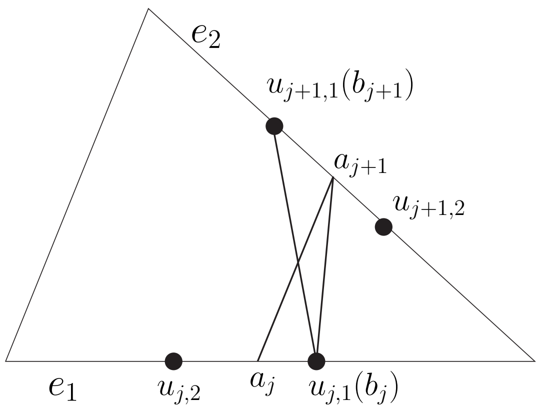


Figure 15: Choosing Steiner points for discrete path

**Proof** We first prove the case when $b_j$ and $b_{j+1}$ are Steiner points. Suppose $b_j$ is between $u_{j,1}$ and $u_{j,2}$ on boundary edge $e_1$ and $b_{j+1}$ is between $u_{j+1,1}$ and $u_{j+1,2}$ on boundary edge $e_2$, as shown in Figure 15. Without loss of generality, suppose $b_j = u_{j,1}$ and $b_{j+1} = u_{j+1,1}$. Let $r$ be the region incident to both $e_1$ and $e_2$. We have $\tau(a_j, a_{j+1}) + \tau_r(a_{j+1}, u_{j+1,1}) \geq \tau(a_j, u_{j+1,1})$ following Proposition 2. As

$$\tau_r(a_{j+1}, u_{j+1,1}) \leq \tau_r(u_{j+1,2}, u_{j+1,1}) \leq \epsilon\tau_{min}(u_{j+1,1})$$

and

$$\tau(a_{j+1}, u_{j+1,1}) \geq \tau_{min}(u_{j+1,1}),$$

we have $\tau(a_j, a_{j+1}) \geq (1-\epsilon)\tau_{min}(u_{j,1})$. We can also prove that $\tau(a_j, a_{j+1}) \geq (1-\epsilon)\tau_{min}(u_{j+1,1})$ in a similar manner. Applying Proposition 2 one more time, we can get the following inequalities:

$$\begin{aligned}
\tau(b_j, b_{j+1}) &= \tau(u_{j,1}, u_{j+1,1}) \\
&\leq \tau_r(u_{j,1}, a_j) + \tau(a_j, a_{j+1}) + \tau_r(a_{j+1}, u_{j+1,1}) \\
&\leq \tau_r(u_{j,1}, u_{j,2}) + \tau(a_j, a_{j+1}) + \tau_r(u_{j+1,2}, u_{j+1,1}) \\
&\leq \epsilon\tau_{min}(u_{j,1}) + \tau(a_j, a_{j+1}) + \epsilon\tau_{min}(u_{j+1,1}) \\
&\leq \frac{2\epsilon}{1-\epsilon}\tau(a_j, a_{j+1}) + \tau(a_j, a_{j+1}) \\
&\leq (1+3\epsilon)\tau(a_j, a_{j+1}),
\end{aligned}$$

since $\epsilon \leq \frac{1}{3}$.

The other two cases can be proved similarly. ∎

With this lemma, we are ready to establish the following theorem:

**Theorem 6** *For any piecewise linear path $p$, there exists a discrete path $p'$ such that $d(p') \leq (1+9\epsilon)d(p)$.*

**Proof** Among the bending points $b_1, b_2, \cdots, b_{k-1}, b_k$ of $p'$, let $b_{i_1}, b_{i_2}, \cdots, b_{i_d}$ be vertices of the triangular decomposition, where $1 = i_1 < i_2 < \cdots < i_{d-1} < i_d = k$. Let $b_{i_j} = v_j$ for $1 \leq j \leq d$. We have

$$\begin{aligned} d(p') &= \sum_{j=1}^{k} \tau(b_j, b_{j+1}) \\ &\leq (1+3\epsilon)\sum_{j=1}^{k} \tau(a_j, a_{j+1}) + \epsilon \cdot (R_{v_1} + 2\sum_{j=2}^{d-1} R_{v_j} + R_{v_d}) \\ &= (1+3\epsilon)d(p) + \epsilon \cdot (R_{v_1} + 2\sum_{j=2}^{d-1} R_{v_j} + R_{v_d}). \end{aligned}$$

For each $j$, $1 \leq j < d$, let $C_j = \sum_{i=i_j}^{i_{j+1}-1} \overline{a_j a_{j+1}}$. That is, $C_j$ is the sub-path of $p$ between $a_{i_j}$ and $a_{i_{j+1}}$. Let $r$ be the region incident to $b_{i_j}$, $a_{i_j}$ and $a_{i_j+1}$ and let $r'$ be the region incident to $b_{i_{j+1}}$, $a_{i_{j+1}-1}$ and $a_{i_{j+1}}$. As $\overline{b_{i_j} a_{i_j}} + C_j + \overline{a_{i_{j+1}} b_{i_{j+1}}}$ is a path from vertex $v_j$ to $v_{j+1}$, we have

$$\tau_r(b_{i_j}, a_{i_j}) + d(C_j) + \tau_{r'}(a_{i_{j+1}}, b_{i_{j+1}}) \geq R_{v_j}$$

as well as

$$\tau_r(b_{i_j}, a_{i_j}) + d(C_j) + \tau_{r'}(a_{i_{j+1}}, b_{i_{j+1}}) \geq R_{v_{j+1}}.$$

As $\tau_r(b_{i_j}, a_{i_j}) \leq \epsilon R_{v_j}$ and $\tau_{r'}(a_{i_{j+1}}, b_{i_{j+1}}) \leq \epsilon R_{v_{j+1}}$, we can get $d(C_j) \geq (1-2\epsilon)(R_{v_j} + R_{v_{j+1}})/2$ and therefore

$$R_{v_1} + 2\sum_{j=2}^{d-1} R_{v_j} + R_{v_d} = \sum_{j=1}^{d-1}(R_{v_j} + R_{v_{j+1}}) \leq \sum_{j=1}^{d-1} \frac{2}{1-2\epsilon} d(C_j) = \frac{2}{1-2\epsilon} d(p) \leq 6d(p).$$

Hence $d(p') \leq (1+3\epsilon)d(p) + 6\epsilon d(p) = (1+9\epsilon)d(p)$. ∎

Combining Theorem 5 and 6 we have the second main result of this paper:

**Theorem 7** *An $\epsilon$-good approximation of the optimum path in regions with flows can be computed in $O(\frac{C_{skew} \cdot n}{\epsilon}(\log \frac{C_{skew}}{\epsilon} + \log n) \log \frac{C_{skew}}{\epsilon})$ time, where $n$ is the number of regions.*

## 6 Conclusion

While we have provided the first decision procedure for the two-dimension flow path problem, the complexity of our algorithm appears far too high to have practical use. There is no known lower bound for this $2\mathcal{D}$ version of the flow path problem. Furthermore, there is no decision algorithm for the $3\mathcal{D}$ version of the flow path problem. It remains an open problem to determine a more exact complexity bound for the $2\mathcal{D}$ and $3\mathcal{D}$ flow path problems.

## References

[1] L. Aleksandrov, M. Lanthier, A. Maheshwari, and J.-R. Sack. An $\epsilon$-approximation algorithm for weighted shortest paths on polyhedral surfaces. In *Proceedings of the 6th Scandinavian Workshop on Algorithm Theory*, volume 1432 of *Lecture Notes in Computer Science*, pages 11–22, 1998.

[2] L. Aleksandrov, A. Maheshwari, and J.-R. Sack. Approximation algorithms for geometric shortest path problems. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 286–295, 2000.

[3] L. Aleksandrov, A. Maheshwari, and J.-R. Sack. An improved approximation algorithm for computing geometric shortest paths. In *Proceedings of the 14th International Symposium on Fundamentals of Computation Theory*, volume 2751 of *Lecture Notes in Computer Science*, pages 246–257, 2003.

[4] J. Canny. Some algebraic and geometric computations in PSPACE. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 460–467, 1988.

[5] J. Canny and J. H. Reif. New lower bound techniques for robot motion planning problems. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pages 49–60, 1987.

[6] G. E. Collins. Quantifier elimination for real closed fields by cylindric algebraic decomposition. In *Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 134–183, 1975.

[7] M. Lanthier, A. Maheshwari, and J. Sack. Approximating weighted shortest paths on polyhedral surfaces. *Algorithmica*, 30(4):527–562, 2001.

[8] M. Lanthier, A. Maheshwari, and J.-R Sack. Shortest anisotropic paths on terrains. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, volume 1644 of *Lecture Notes in Computer Science*, pages 524–533, 1999.

[9] C. Mata and J. Mitchell. A new algorithm for computing shortest paths in weighted planar subdivisions. In *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, pages 264–273, 1997.

[10] J. S. B. Mitchell. Geometric shortest paths and network optimization. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 633–701. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.

[11] J. S. B. Mitchell and C. H. Papadimitriou. The weighted region problem: Finding shortest paths through a weighted planar subdivision. *Journal of the ACM*, 38(1):18–73, January 1991.

[12] N. Papadakis and A. Perakis. Minimal time vessel routing in a time-dependent environment. *Transportation Science*, 23(4):266–276, 1989.

[13] N. Papadakis and A. Perakis. Deterministic minimal time vessel routing. *Operations Research*, 38(3):426–438, 1990.

[14] C. H. Papadimitriou. An algorithm for shortest-path motion in three dimensions. *Information Processing Letters*, 20:259–263, 1985.

[15] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, pages 421–427, 1979.

[16] J. H. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science*, pages 144–154, 1985.

[17] J. H. Reif and J. A. Storer. A single-exponential upper bound for finding shortest paths in three dimensions. *Journal of the ACM*, 41(5):1013–1019, September 1994.

[18] J. H. Reif and Z. Sun. An efficient approximation algorithm for weighted region shortest path problem. In *Proceedings of the 4th Workshop on Algorithmic Foundations of Robotics*, pages 191–203, 2000.

[19] J. H. Reif and Z. Sun. On frictional mechanical systems and their computational power. *SIAM Journal on Computing*, 32(6):1449–1474, 2003.

[20] J. T. Schwartz and M. Sharir. On the piano movers problem: II. general techniques for computing topological properties of real algebraic manifolds. *Advances in applied mathematics*, 4:298–351, 1983.

[21] J. Sellen. Direction weighted shortest path planning. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, pages 1970–1975, 1995.

[22] Z. Sun and J. H. Reif. BUSHWHACK: An approximation algorithm for minimal paths through pseudo-Euclidean spaces. In *Proceedings of the 12th Annual International Symposium on Algorithms and Computation*, volume 2223 of *Lecture Notes in Computer Science*, pages 160–171, 2001.

[23] Z. Sun and J. H. Reif. Adaptive and compact discretization for weighted region optimal path finding. In *Proceedings of the 14th International Symposium on Fundamentals of Computation Theory*, volume 2751 of *Lecture Notes in Computer Science*, pages 258–270, 2003.

[24] Z. Sun and J. H. Reif. On energy-minimizing paths on terrains for a mobile robot. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, pages 3782–3788, 2003.

[25] G. Wilfong. Motion planning in the presence of movable obstacles. In *Proceedings of the 4th Annual ACM Symposium on Computational Geometry*, pages 279–288, 1988.