Acta Informatica 26, 93–122 (1988)

Motion Planning Among Time Dependent Obstacles *

K. Sutner [1] and W. Maass [2]
[1] Stevens Institute of Technology, Hoboken, NJ 07030, USA
[2] University of Illinois at Chicago, Chicago, IL 60638, USA

Summary. In this paper we study the problem of motion planning in the presence of time dependent, i.e. moving, obstacles. More specifically, we will consider the problem: given a body B and a collection of moving obstacles in D-dimensional space decide whether there is a continuous, collision-free movement of B from a given initial position to a target position subject to the condition that B cannot move any faster than some fixed top-speed c. As a discrete, combinatorial model for the continuous, geometric motion planning problem we introduce time-dependent graphs. It is shown that a path existence problem in time-dependent graphs is PSPACE-complete. Using this result we will demonstrate that a version of the motion planning problem (where the obstacles are allowed to move periodically) is PSPACE-hard, even if $D=2$, B is a square and the obstacles have only translational movement. For $D=1$ it is shown that motion planning is NP-hard. Furthermore we introduce the notion of the c-hull of an obstacle: the c-hull is the collection of all positions in space-time at which a future collision with an obstacle cannot be avoided. In the low-dimensional situation $D=1$ and $D=2$ we develop polynomial-time algorithms for the computation of the c-hull as well as for the motion planning problem in the special case where the obstacles are polyhedral. In particular for $D=1$ there is a $O(n \lg n)$ time algorithm for the motion planning problem where n is the number of edges of the obstacle.

1. Introduction and Definitions

The classical piano movers' problem is to plan the movement of a body B in two or three dimensions from a given initial to a target position avoiding

Offprint requests to: K. Sutner

* During the preparation of this paper the second author has been supported in part by NSF grant DCR-85-04247

collision with an obstacle A that remains fixed throughout the movement. A variety of motion planning problems in various dimensions have been studied extensively, see e.g. [SS1, SS2, R]. The moving object B may be a disk, a rigid polygon, many jointed (see [R, HJW]) or indeed consist of several independent parts as in [SS3, SY, Y]. In all these situations the obstacles are assumed to be static, i.e. they do not change position or shape as the object moves from its initial to its final position. For any fixed B and sufficiently simple obstacles polynomial time algorithms are known for the motion planning problem. It becomes intractable if the moving object is allowed to have unbounded degree of freedom; e.g. it is shown in [HSS] that the coordinated motion planning problem for arbitrarily many rectangular bodies amidst polygonal obstacles in PSPACE-hard, see also [R, SY].

In this paper we will study motion planning in the presence of time dependent obstacles. We restrict our attention to the "robot navigation" problem where an object B (the robot) moves autonomously among the obstacles. Furthermore we will assume that the speed of B cannot exceed some arbitrary but fixed constant $c>0$. The problem of steering a vehicle in the midst of other moving vehicles provides a typical example for this type of motion planning problem. In this case the obstacles are themselves physical bodies with fixed shape and move along certain paths. We do not wish to restrict ourselves to this situation through: more generally a time dependent obstacle may be viewed as any constraint on the set of allowed positions of B that varies in time. A traffic light for example can be modelled by a forbidden line that occurs periodically. Our model assumes complete information, i.e. all the obstacles are completely specified and thus all the forbidden positions are known from the beginning.

One should notice that a dynamic motion planning problem with time dependent obstacles involving D spatial dimensions is quite different from a $D+1$-dimensinal classical movers' problem with static obstacles: movement along the additional time axis is forced and irreversible. This leads to a new phenomenon: B may be in a position where it does not even touch an obstacle (a so called free position), yet it is doomed to collide with it at some later moment. The collection of all such positions will be called the c-hull of the obstacle. The boundary of the c-hull corresponds to the manifold of semi-free positions in the classical problem: it is the set of all positions where a collision can just barely be avoided.

There are two basic possibilities concerning the descriptions of the obstacles. First one can specify an obstacle in two pieces:

– its geometrie shape, expressed as a D-dimensional subset of space,
– its placement in space as a function of time.

Alternatively one can describe a moving obstacle as a $D+1$-dimensional pointset in space-time. This includes changes in the shape as well as the location of an obstacle. We will give algorithms for the special case where the obstacles the polytopes (for $D=1$, 2) that are polynomial in the number of vertices of these polytopes. The first approach is of interset in particular for periodic movements that occur e.g. in rotating objects. It is rather difficult to measure the complexity of the corresponding point set in space-time: even for very simple rotational movements this set fails in general to be algebraic.

As one might expect, the problem becomes intractable if one allows a concise description of obstacles in terms of their shape and some periodic movement. In order to prove PSPACE-hardness we will introduce a combinatorial, discrete analogue of the geometric, continuous motion planning problem. We will refer to these combinatorial analogues as time dependent graphs (td-graphs). A td-graph models the following simple case of a motion planning problem with time dependent obstacles:

- only finitely many locations exist,
- some of these locations are periodically occupied by an obstacle,
- movement between locations is limited and carries a time cost.

Using td-graphs our PSPACE-hardness proof proceeds in two steps: first we will simulate linear bounded automata acceptance by a path existence problem for td-graphs. The crucial idea in the simulation is to encode tape inscriptions by time. A configuration of the machine then corresponds to a vertex in a td-graph and a computation can be construed as a path in that graph. Acceptance is thus reduced to the existence of a path from a certain source vertex to a target vertex. In a second step one can then simulate path existence in td-graphs by a concrete motion planning problem in two dimensions. The obstacles will have very simple rectangular shape and either be stationary or only move parallel to the axes. In particular no rotational movement will be required.

The discrete problem is also of interest in its own sake: motion planning for a train in a railroad network for example is readily expressed as a reachability problem for a td-graph. We hope to present a more thorough discussion of td-graphs in a future paper.

As an underlying computational model will use a *real RAM*, i.e. a random access machine that can handle real numbers as basic objects and has primitive operations such as k-th roots and trigonometric functions available at unit-cost.

The paper is organized as follows: in Sect. 1 we give the necessary definitions and derive some basic properties of the c-hull and the collection of reachable points. Then we introduce td-graphs in Sect. 2 and prove PSPACE-completeness of a suitable path existence problem in td-graphs. Section 3 is devoted to a simulation of the discrete problem by a dynamic motion planning problem in two dimensions. This establishes PSPACE-hardness of a very simple form of the motion planning problem. For the one-dimensional problem we prove NP-hardness. Lastly in Sect. 4 we give polynomial time algorithms for the one-dimensional situation – which unlike the classical movers' problem is not trivial – in the case where the obstacles are given as polyhedra in space-time. We also give an outline of the solution for the two-dimensional case.

Upon completion of a draft of this paper we learned that J. Reif and M. Sharir had independently achieved similar results at about the same time; see [RS]. In particular their paper contains a direct simulation of linear bounded automaton acceptance in terms of a motion planning problem in three-dimensions.

Motion planning problems are most conveniently expressed and studied in space-time. For our purposes space-time is the $D+1$-dimensional real vectorspace $\mathbf{R}^D \times \mathbf{R}$. Space-time will be construed as having D space dimensions and one additional time dimension. A point p in space-time is a position; thus $p=(x, t)\in\mathbf{R}^D\times\mathbf{R}$ where x is the location of p. Let $T((x, t)):=t$ be the natural projection from space-time into time. For two positions p and q ray(p, q) denotes the one-sided infinite ray that starts at p and passes through q; $[p, q]$ denotes the line segment with endpoints p and q. $K_\varepsilon(p)$ denotes the closed sphere of radius ε around p.

Let $H_\tau:=\{(x, \tau)\,|\,x\in\mathbf{R}^D\}$ be the hyperplane perpendicular to the t-axis with t-axis intercept $t=\tau$. For any subset X of space time let $X(\tau):=X\cap H_\tau$ be the projection of X at time τ. As usual clos(A) will be the closure of A, int(A) its interior and ∂A its boundary. $S^D:=\partial K_1(0)\subset\mathbf{R}^{D+1}$ stands for the surface of the unit sphere in space-time. For $e\in S^D$ let ray$(p\!:\!e)$ be the ray starting at p in direction e, thus ray$(p\!:\!e)=\{p+\alpha e\,|\,\alpha\geqq 0\}$.

Lastly let $\mathbf{N}$ be the set of natural numbers and N^+ the set of positive natural numbers. For $n\in\mathbf{N}$ set $[n]:=\{1, 2, \ldots, n\}$.

Obstacles and the Moving Body

Throughout this paper we will not allow B to rotate. Furthermore we will insist that B is a D-dimensional closed polyhedron. A *(compound) obstacle* is a finite collection $\mathbf{A}=A_1, \ldots, A_m$ of atomic obstacles; an *atomic obstacle* is a $(D+1)$-dimensional closed polyhedral subset of space-time. An atomic obstacle is described either

- explicitly as a polyhedral pointset in space-time or
- implicitly as a D-dimensional compact polytope C that represents the shape of the moving object plus a description of its movement.

Again we disallow any rotational movement. An implicitly defined obstacle will frequently be identified with its *trajectory*, i.e. the set of positions in space-time occupied by it. Note that this trajectory forms a closed subset of space-time. We do not require the trajectories of different atomic obstacles in a compound obstacle to be disjoint.

c-Paths

A path P is a curve in space-time parametrized by time; thus $P\colon I\to\mathbf{R}^{D+1}$ where $I\subset\mathbf{R}$ is a possibly infinite interval and $T(P(t))=t$. It may be construed as the world-line of a particle moving through space. Throughout this paper a path will always be continuous and piecewise continuously differentiable. Now fix a positive constant c once and for all. c denotes the maximum speed of the moving object B (though not of the time-dependent obstacles). A path P is called a *c-path* iff the speed of a particle whose world-line is P never exceeds c.

Note that the movement expressed by a c-path is not required to be smooth (i.e. speed may change discontinuously).

As body B is restricted to purely translational movement and not allowed to rotate its movement in space-time can be completely described by

- a fixed orientation B and one fixed base point of B
- the path for trajectory of the base point (which is of course a c-path).

Let us agree that $B \subset \mathbf{R}^D$ is given in the proper orientation and contains the origin which will serve as base point. We let $B(p)$ denote the set of positions occupied by B when placed at position p; thus $p \in B(p) = \{p + b \mid b \in B\}$.

Implicitly defined obstacles are handled in the same way. However, unlike with body B obstacles will be allowed to disappear and reappear. Therefore the path for an obstacle may have as domain a collection of closed intervals $I = \bigcup I_i$.

We will need a little bit more terminology. Let $\gamma := \operatorname{atn}(c)$ be the maximum slope of a c-path. For a straight line L in space-time let its inclination $\operatorname{incl}(L)$ be the angle between L and the time-axis, $0 \leqq \operatorname{incl}(L) \leqq \pi/2$. For an affine subspace K let $\operatorname{incl}(K) := \inf(\operatorname{incl}(L) \mid L \subset K)$. Also let $S_c := \{e \in S^D \mid \operatorname{incl}(e) \leqq \gamma\}$.

Lastly define $\operatorname{cone}(p) := \operatorname{cone}_+(p) \cup \operatorname{cone}_-(p)$ where

$$\operatorname{cone}_+(p) := \bigcup \{\operatorname{ray}(p : e) \mid e \in S_c\} \quad \text{and} \quad \operatorname{cone}_-(p) := \bigcup \{\operatorname{ray}(p : -e) \mid e \in S_c\}.$$

Observe that for any c-path $P: \mathbf{R} \to \mathbf{R}^{D+1}$ and any time τ the whole path P is contained in $\operatorname{cone}(P(\tau))$.

Free Positions & Admissible Paths

Now let $\mathbf{A}$ be a compound obstacle, $\mathbf{A} = A_1, \ldots, A_m$, and B a body to be moved about. A position p in space-time is called *free* with respect to A iff $B(p)$ does not intersect any of the $A_1, \ldots, A_m$. A position p in space-time is called *semi-free* with respect to $\mathbf{A}$ and B iff $B(p)$ does not intersect the interior of any of the $A_1, \ldots, A_m$. A position that fails to be semi-free is called forbidden. A c-path is said to be $\mathbf{A}$, B-admissible (or simply to avoid $\mathbf{A}$) iff P contains no forbidden positions with respect to $\mathbf{A}$ and B.

Hence if B moves along a path that avoids $\mathbf{A}$ it may touch the obstacle but will not collide with it. Using the "inflate-and-shrink" method described in [LPW] one can expand the obstacles and shrink B to a point B_0. The inflated obstacle $\mathbf{A}_B$ has the same properties with respect to motion planning for B_0 that $\mathbf{A}$ has with respect to B: c-path P is A, B-admissible iff P is $\mathbf{A}_B$, B_0-admissible iff P does not intersect the interior of $\mathbf{A}$. Note that $\mathbf{A}_B$ is again a polyhedral set of positions in space-time.

To simplify notation we will assume from now on that, unless explicitly stated otherwise, B is a point.

One way to determine whether B can move from some position p to p' avoiding obstacle $\mathbf{A}$ is to compute the set of all positions that can be reached from p:

$$\mathbf{R}(\mathbf{p}; \mathbf{A}) := \{q \mid \text{there is an } \mathbf{A}\text{-asmissible } c\text{-path from } p \text{ to } q\}.$$

Then p' can be reached from p iff $p' \in R(p, A)$. For the empty obstacle $A = \emptyset$ clearly $R(p; \emptyset) = \text{cone}_+(p)$. See Sect. 4 for an explicit computation of $R(p; \mathbf{A})$ for polyhedral obstacles using plane and space sweep techniques.

The c-Hull

Consider the following rudimentary form of a motion planning problem: Given an obstacle **A** and a source position $p = (x, t)$ is it possible for B to start moving at position p and avoid collision with **A** indefinitely?

In other words, is there an **A**-admissible c-path P: $\{\tau \in \mathbf{R} \mid \tau \geqq t\} \to \mathbf{R}^{D+1}$. Any such a path will be called an escape-path from p. The collection of points that fail to have an escape-path will be called the *c-hull* of **A**:

$$c\text{-hull}(A) := \{p \in \mathbf{R}^{D+1} \mid \text{there is no escape-path from } p\}.$$

Figure 1 shows the c-hull of an obstacle in one dimension for $c = 1$. Notice that for infinite speed $c = \infty$ the c-hull of a convex open obstacle coincides with the (interior of the) obstacle.

One can use time-inversion to define dual notions like co-reachability, co-$R(p; \mathbf{A}) = \{q \mid p \in R(q; \mathbf{A})\}$, so-escape-path, co-$c$-hull and so forth. A position in the co-c-hull for example can be reached only from positions within the co-c-hull. The arguments for all these dual notions are exactly the same as for the ordinary notions and will therefore be omitted.

Suppose the top-speed c is fixed once and for all. There is a variety of decision and search problems associated with motion planning in the presence

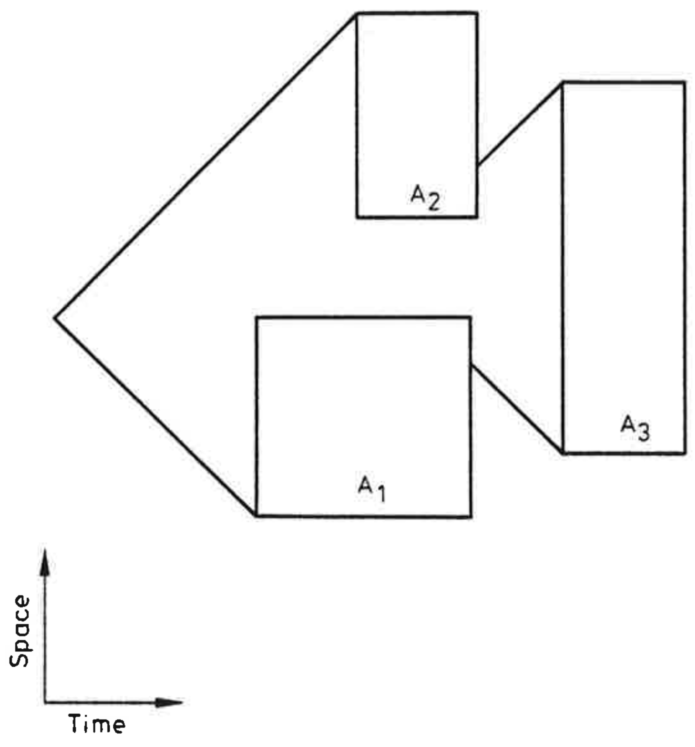


Fig. 1. The c-hull of a compound obstacle in one dimension

of time dependent obstacles. We will here focus on reachability problems: given an obstacle **A**, a source position $p=(x, t)$ and a target position $p'=(x', t')$ decide whether there is an admissible c-path from p to p'. Similarly one might specify only a target location x' and ask whether for some $t' \geqq t$ there is an admissible c-path to $p' := (x', t')$. A deadline T can be imposed so that one has to determine the existence of an admissible c-path to $p' := (x', t')$ for some time t', $t \leqq t' \leqq T$. Formally these motion planning problems are defined as follows.

Position-to-Position Reachability Problem

Instance: An obstacle **A**, source and target positions $p=(x, t)$ and $p'=(x', t')$.
Question: Is there a c-path from p to p' that avoids obstacle **A**?

Position-to-Location Reachability Problem

Instance: An obstacle **A**, a source position $p=(x, t)$ and a target location x'.
Question: Is there a time $t' \geqq t$ and a c-path from p to $p'=(x', t')$ that avoids obstacle **A**?

Position-to-Location Reachability Problem with Deadline

Instance: An obstacle **A**, a source position $p=(x, t)$, a target location x' and a deadline $T \geqq t$.
Question: Is there a time t', $t \leqq t' \leqq T$, and a c-path from to $p'=(x', t')$ that avoids obstacle **A**?

Beyond deciding whether a position or location can be reached one would naturally like to construct an appropriate c-path. For example one might wish to construct the shortest or safest c-path leading from one position to another. We will not address these problems at this point. However, we will study the problem of computing the c-hull:

Hull Problem

Instance: An obstacle **A**.
Output: The c-hull of **A**.

Note that a solution to the Hull Problem also provides a solution to the Escape Problem which is to determine whether there exist an escape path from a given position.

In the remainder of this section we will derive simple topological properties of the set of reachable points and the c-hull that are implicit in our definitions. For example the fact that we allow c-paths to have speed less than or equal to c (rather than just less) and our choice of obstacles and B as closed sets have the consequence that $R(p; \mathbf{A})$ is a closed set in space-time. The results in this section do not use our assumption that **A** is a collection of polyhedral sets.

Lemma (1.1). *Let* **A** *be an obstacle, p a position. Then* $R(p; \mathbf{A})$ *is a closed set in space-time.*

Lemma (1.2). *Let* **A** *be an obstacle. Then the c-hull of* **A** *is an open set in space-time.*

Proof. We will only show that $R(p; \mathbf{A})$ is closed. the argument can easily be modified to provide a proof of Lemma 1.2. Let $p=(x_0, t_0)$ be the given position and assume that $(q_j)_{j \geqq 0}$ is a sequence of positions in $R(p; A)$ that converges against some position $q=(x, t)=\lim_{j \to \infty} q_j$. It is safe to assume that $p \neq q$, whence $t > t_0$. For every $j \geqq 0$ pick a c-path P_j from p to q_j that avoids obstacle $\mathbf{A}$. In general there are globally different c-paths from a source to a target position, therefore the sequence $(P_j)_{j \geqq 0}$ may fail to converge. However, by the theorem of Arzela-Ascoli (see e.g. [DS, IV.5.7]) there is a uniformly convergent subsequence $(P_{j_i})_{i \geqq 0}$. Define a path $Q\colon [t_0, t] \to \mathbf{R}^{D+1}$ by $Q := \lim_{i \to \infty} P_{j_i}$.

Then Q is a c-path, $Q(t_0)=p$ and $Q(t)=q$. Furthermore Q must avoid $\mathbf{A}$, for otherwise we would have that P_{j_i} intersects $\operatorname{int}(\mathbf{A})$ for some $i \geqq 0$, contradicting the fact that P_{j_i} is $\mathbf{A}$-admissible. Thus q is reachable from p and we are done. □

The following lemma shows that escaping from one convex obstacle is either trivial or impossible: if there is an escape-path at all, then there actually is a straight escape-path of the form $\operatorname{ray}(p\colon e)$ for some $e \in S_c$.

Lemma (1.3). *Let* $\mathbf{A}$ *be a convex obstacle and* p *a position such that there exists an escape-path from* p. *Then there actually is a straight escape-path from* p.

Proof. Case 1: $\mathbf{A}$ is bounded in time.

Let $t_1 := \max(T(q) \mid q \in \mathbf{A}) + 1$. Let Q be an escape-path starting at $p_0=(x_0, t_0)$ that exists by our assumption, $Q\colon \{\tau \in \mathbf{R} \mid \tau \geqq t_0\} \to \mathbf{R}^{D+1}$. For q on Q, $q \neq p_0$, consider the ray $\operatorname{ray}(p_0, q)$. We will show that for some q $\operatorname{ray}(p_0, q)$ does not intersect the interior of obstacle $\mathbf{A}$.

For the sake of a contradiction suppose otherwise. Define two maps:

$$f, g\colon \{\tau \in \mathbf{R} \mid \tau > t\} \to \mathbf{R}^{D+1},$$

$$f(\tau) := \text{first point in } \operatorname{ray}(p, Q(\tau)) \cap \mathbf{A},$$

$$g(\tau) := \text{last point in } \operatorname{ray}(p, Q(\tau)) \cap \mathbf{A}.$$

See Fig. 2 for an illustration. Both f and g are well-defined as $\operatorname{ray}(p, Q(\tau)) \cap A$ is closed and bounded. For all $\tau > t_0$ we have $T(f(\tau)) \leqq T(g(\tau))$. Also observe that $\tau < T(f(\tau))$ for τ very close to t_0 and $T(g(t_1)) < t_1$. Hence by continuity there is a t', $t_0 < t' < t_1$, such that $T(f(t')) \leqq t' \leqq T(g(t'))$.

Now set $q' := Q(t')$. Note that $\operatorname{ray}(p_0, q')$ is a c-path. To obtain a contradiction it thus suffices to show that $\operatorname{ray}(p_0, q')$ is admissible, i.e. it fails to intersect $\operatorname{int}(A)$. This follows from the next claim.

Claim. $\operatorname{ray}(p, q')$ is tangent to A.

Proof of claim: First let $q_1 := f(t')$ and $q_2 := g(t')$. By the definition of f, g and the convexity of $\mathbf{A}$ we have $[q_1, q_2] = \operatorname{ray}(p_0, q') \cap \mathbf{A}$. Now suppose $\operatorname{ray}(p_0, q')$ intersects $\operatorname{int}(\mathbf{A})$, say in point $r \in [q_1, q_2]$. We may assume without loss of generality that q' lies between q_1 and r, the other case being entirely similar. Then for some positive ε $K_\varepsilon(r) \subset \operatorname{int}(\mathbf{A})$. By the convexity of $\mathbf{A}$ the cone with apex q_1 generated by the line segments $[q_1, r']$ as r' ranges over $K_\varepsilon(r)$ is a subset of $\mathbf{A}$. But q' is contained in the interior of this cone, hence q' lies in the interior of $\mathbf{A}$, contradiction.

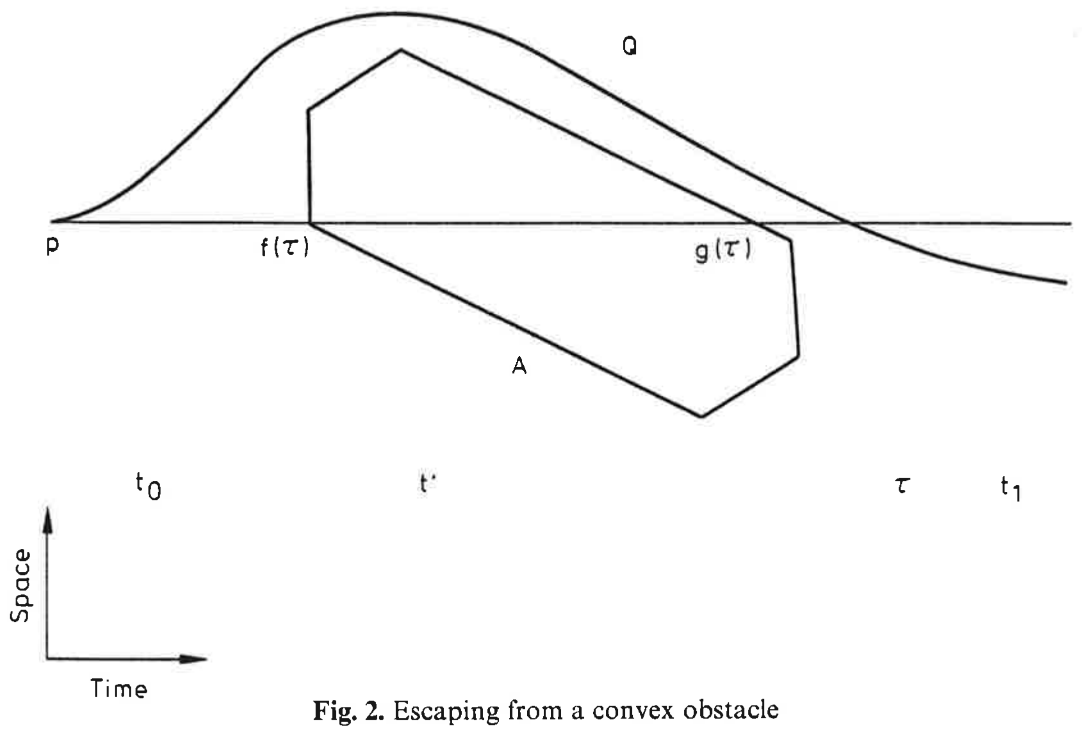


Fig. 2. Escaping from a convex obstacle

Case 2: **A** unbounded in time.

Assume for the sake of a contradiction that there is no straight escape path. Define a map

$$f\colon\ S_c \to \mathbf{R}^{D+1} \quad \text{by} \quad f(e) := \text{the first point in ray}(p\colon e) \cap \mathbf{A}.$$

$f(e)$ exists by our assumption and is clearly continuous. Recall that S_c is compact, so $f[S_c]$ is compact. Now let $t_1 := \max(T(f[S_c]))$ and truncate **A** at time $t_1 + 1$. The truncated obstacle is again convex as the intersection of two convex sets, it is bounded in time and does not admit any straight escape paths. This contradicts case 1. □

Using Lemma 1.3 we will show in a moment that the c-hull of a convex obstacle is again convex: it is the intersection of all open half-spaces whose boundary hyper-planes contain escape-paths. Note that contrary to the classical movers' problem one now has to deal with the difficulty that some of the surface patches of **A** may not contain any c-paths (even if the surface patch is planar).

Consider for example in one dimension an obstacle of length 1 that moves perpetually at speed, say, $2c$. Then **A** is a strip in space-time whose boundaries are parallel lines L_1 and L_2 of slope larger than γ. Suppose L_1 precedes L_2 with respect to time. Then every admissible c-path that contains a point of L_1 ends there. Hence the c-hull of **A** is the half-plane defined by L_2 that contains L_1.

For a point p on the surface of **A** to lie on the boundary of the c-hull it is clearly necessary that locally around p there is a c-path starting at p. We will show in Theorem 1.4 that this condition is also sufficient if **A** is convex.

To this end suppose **A** is a convex obstacle. A hyper-plane H that does not intersect the interior of **A** is said to avoid **A**. For any hyper-plane H that avoids A let us introduce the following terminology. H_- denotes the open half-space defined by H that contains int(**A**) and H_+ is the complement of H_-, $H_+ := \mathbf{R}^{D+1} - H_-$. Thus $H_+ \cap \mathbf{A} \subset \partial\mathbf{A}$. Also let e_H be the unit normal vector for H oriented in such a fashion that, for all p on H, the point $p+e_H$ lies in H. H will be called posterior iff $T(e_H) > 0$.

A typical example of a posterior plane is $H_\tau = \{(x, \tau) \mid x \in \mathbf{R}^D\}$ provided that **A** contains no positions with time component larger than τ. The next proposition follows readily from the definition of a posterior hyper-plane.

Proposition. *Let H be a hyper-plane that avoids A and $p=(x, t)$ a position in H_+. Then:*

(1) *H posterior implies that* ray$(p\!:\!0)$ *in an escape-path for p*

(2) *if H is not posterior then for $t_1 \geqq t$: $p_1 := (x, t_1)$ lies on H.*

Theorem (1.4). *Let **A** be a convex obstacle. Then the c-hull of **A** is the intersection of all open half-spaces H_- that contain its interior and whose boundary hyper-plane H has inclination at most γ or in posterior:*

$$c\text{-hull}(\mathbf{A}) = \bigcap \{H_- \mid H \textit{ avoids } \mathbf{A}, H \textit{ posterior or } \mathrm{incl}(H) \leqq \gamma\}.$$

Proof. Let $\mathfrak{Z}$ denote the collection of all the half-spaces as in the statement of the theorem. We will first show that any point $p=(x, t)$ in the c-hull must also lie in $\bigcap \mathfrak{Z}$.

Let H be an arbitrary hyper-plane H in $\mathfrak{Z}$. Suppose for the sake of a contradiction that $p \in H_+$. By (1) of the proposition H cannot be posterior. Hence we must have incl$(H) \leqq \gamma$. Let $p_1 = (x, t_1)$, $t_1 \geqq t$, be a position on H as in part (2) of the proposition. Then there exists an escape-path for p: combine $[p, p_1]$ with any c-path on H starting at p_1; the latter exists as incl$(H) \leqq \gamma$. Hence p is in the c-hull, contradiction.

For the opposite direction suppose $p=(x, t)$ does not lie in the c-hull of **A**. By Lemma 1.3 p has a straight escape-path P of the form ray$(p\!:\!e)$. As **A** and P are both convex there must be a hyperplane H that separates them, i.e. $P \subset H_+$ and int(**A**) $\subset H_-$. This follows from the Hahn-Banach theorem, see [DS, V.1.12]. Suppose H fails to be posterior and in addition incl$(H) > \gamma$. We are heading for a contradiction, so it suffices to show that under these assumptions P must intersect H_-. To see this let us assume, without loss of generality, that the origin lies on H. Thus $H = \{x \in \mathbf{R}^{D+1} \mid x \circ e_H = 0\}$. Furthermore, $P = \{p + \alpha e \mid \alpha \geqq 0\}$ for some $e \in S_c$. Now $(p+\alpha e) \circ e_H = p \circ e_H + \alpha \cdot e \circ e_H$ and $p \circ e_H \geqq 0$ as $p \in H_+$. On the other hand $e \circ e_H < 0$ as H is not posterior and has inclination larger than γ. Thus for sufficiently large $\alpha \geqq 0$ the position $p + \alpha e$ lies in H_- and we are done. □

The following two corollaries show the similarity between c-hull and convex hulls.

Corollary (1.5). *The* c-hull *of a convex obstacle is again convex.*

Corollary (1.6). *If obstacle **A** is convex then the* c-hull *of **A** is the intersection of all open half-spaces H_- where H is a hyper-plane tangent to **A** that is posterior or has inclination at most γ.*

In particular if **A** a convex polyhedron then the c-hull of **A** is again a convex polyhedron. Exactly those faces of **A** that belong to posterior planes or have inclination at most γ are also faces of the c-hull. Our next result provides a description of the c-hull in the general situation in terms of escape-paths.

Lemma (1.7). *Let* **A** *be an obstacle and p a position not in its interior. Then p lies on the boundary of the* c-hull *of* **A** *iff there exists an escape-path from p and every such escape-path has initial speed c. Indeed, there is an escape-path that has an initial segment of the form $[p, p']$ of inclination γ where p' lies on the boundary of* **A**.

Proof. Let $C := c\text{-hull}(\mathbf{A})$ and $p_0 = (x_0, t_0)$ a position on the boundary of C but not in $\operatorname{int}(A)$. By Lemma 1.2 C is open, so p_0 must be have an escape-path $P: \{\tau \in \mathbf{R} \mid \tau \geqq t\} \to \mathbf{R}^{D+1}$. Set

$$I := \{t \geqq t_0 \mid \forall \tau \in [t_0, t]([p_0, P(\tau)] \text{ is admissible})\}.$$

Claim. For all $t \in I$ the inclination of $[p_0, P(t)]$ is equal to γ. As P is a c-path so is $[p_0, P(t)]$, whence $\operatorname{incl}([p_0, P(t)]) \leqq \gamma$. So assume that for some $t_1 \in I$ $\operatorname{incl}([p_0, P(t_1)]) < \gamma$. Then for some point p_2 on $[p_0, P(t_1)]$ sufficiently close to p_0 there is a positive ε such that $K_\varepsilon(p_0) \subset \operatorname{cone}_-(p_2)$. But then every position q in $K_\varepsilon(p_0)$ not in $\operatorname{int}(\mathbf{A})$ has an escape-path: combine $[q, p_2]$ with $[p_2, P(t_1)]$ and the remainder of P.

Hence if I is unbounded then P is really a ray and we are through. If on the other hand I is bounded then there must be a $t_1 \in I$, $t_1 > t_0$, such that $p_1 := P(t_1)$ lies on the boundary of **A**. For otherwise an argument similar to the one used in the proof of the claim could be used to show that for some $\tau > 0$: $\sup(I) + \tau \in I$. As $[p_0, p_1]$ and $\partial\mathbf{A}$ are closed, they intersect in some first point p' and the proof is finished. □

The first position q_p where an escape path for p touches **A** will be called the *escape-point* for p. Note that the escape-point is in general not unique, e.g. the position corresponding to the tip of the c-hull displayed in Fig. 1 has two escape-points associated with it. However, the collection of all positions with more than one escape-point forms a set of measure 0.

The boundary of the c-hull of **A** can now be described as follows: it consists of surface patches of **A** and portions generated by line segments $[p, q_p]$. Again in the special case where **A** is polyhedral and, say, $D=2$ one can show that escape-points are always interior points of the edges of **A** or non-convex vertices. In Chap. 4 we will use this description of the c-hull to given polynomial time algorithms for its construction.

2. Time Dependent Graphs

We now introduce a discrete combinatorial model for motion planning in the presence of time-dependent obstacles. Suppose body B can be placed at only finitely many locations. It is convenient to think of these locations as the vertices of a direct graph H where an edge (x, y) indicates the possibility to move from location x to location y. For an application of this idea to the static motion planning problem see [SS 1]. In a dynamic environment where obstacles move about and the movement of B is subject to a speed limit H can be augmented in the following way. Every edge (x, y) has a positive cost associated with it, corresponding to the amount of time needed to travel from x to y. Furthermore, a vertex may be in state 0 or 1: a vertex in state 0 at time t represents a location that is currently occupied by an obstacle and therefore cannot be used by B. Similarly a vertex in state 1 at time t represents a location can be used by B. The augmented graph will be called a time-dependent graph, or td-graph for short. Motion planning can now be expressed as a path existence problem in a td-graph: one has to find a path from source vertex to a target vertex of prescribed total cost that uses only vertices in state 1. More formally define a td-graph as follows:

Definition. Time-Dependent Graph (*td-graph*). A time-dependent graph $H^\infty = \langle H, \Delta, \sigma \rangle$ consists of:

- a finite direct graph $H = \langle V, E \rangle$, possibly containing self-loops, called the static graph of H^∞,
- an edge labeling $\Delta: E \to N^+$, called the delay function of H^∞,
- a periodic function $\sigma: V \times N \to \{0, 1\}$, called the status function of H^∞. Periodic here means that for some positive integer π and all $t \geqq 0$, $v \in V$: $\sigma(v, t)) = \sigma(v, t+\pi)$.

  H^∞ can be interpreted as a locally finite, infinite directed graph $H^\infty = \langle V^\infty, E^\infty \rangle$ where vertex set V^∞ and edge set E^∞ are defined as follows:

$$V^\infty := \{(x, t) \in V \times N \mid \sigma(x, t) = 1\} \quad \text{and}$$
$$E^\infty := \{((x, t), (y, s)) \in V^\infty \times V^\infty \mid (x, y) \in E \wedge s - t = \Delta(x, y)\}.$$

We will usually write v^t instead of (v, t) for the t-th copy of vertex v in H^∞, $t \geqq 0$. The vertices in H^∞ will be called nodes (to distinguish them from the vertices in H). The least $\pi \geqq 1$ such that $\sigma(v, t) = \sigma(v, t+\pi)$ for all $t \geqq 0$, $v \in V$, is called the period of H.

With a view towards the Position-to-Location motion planning problem introduced in the last section let us define the following node-to-vertex path existence problem for td-graphs:

Path Existence for Time-Dependent Graphs (*PETD*)

Instance: A td-graph $H^\infty = \langle H, \Delta, \sigma \rangle$, a source node x^s in V^∞ and a target vertex y in V.

Question: Is there a directed path in H^∞ from x^s to y^t for some $t \geqq s$?

Similarly one can introduce note-to-node and note-to-vertex with deadline path existence problems. For our purposes here the node-to-vertex version is most convenient, therefore we will focus on PETD as defined above. All our arguments are easily modified to deal with the other two versions. To assess the computational complexity of PETD one has to agree on a way of coding the status function. We will here assume that σ is given as list $\sigma = \langle \sigma_v : v \in V \rangle$ of component functions $\sigma_v : \mathbf{N} \to 2$ with the understanding that $\sigma(v, t) = \sigma_v(t)$. Component function σ_v in turn is represented by a simple on-line program using only statements of the following form:

$$z_1 := a$$
$$z_1 := z_2$$
$$z_1 := z_2 + z_3$$
$$z_1 := z_2 - z_3$$
$$z_1 := z_2 \textbf{ mod } a$$
$$z_1 := \textbf{if } z_2 = 0 \textbf{ then } z_3 \textbf{ else } z_4.$$

Here $a \in \mathbb{N}$ is a constant and all the variables z_i are supposed to range over the natural numbers. Clearly $\sigma_v(t)$ can be computed in time polynomial in the size of σ_v and $\lg t$ (for our purposes it actually suffices to insist that $\sigma_v(t)$ be computable in polynomial space).

The size of td-graph H^∞ is then the total number of bits needed to specify H, Δ and σ and thus $O(n^2 \cdot \lg \Delta_0 + n \cdot L \cdot \lg K)$ where $n := |V|$, $\Delta_0 := \max(\Delta(e) | e \in E)$, L in the maximum number of instructions in any of the programs describing σ_v, $v \in V$, and K is the largest constant occuring in these instructions. Thus $\sigma_v(t)$ can be computed in $O((L + \lg K + \lg t)^2)$ steps. Also note that a node v^t of H^∞ can be specified in $O(\lg n + \lg t)$ bits.

Hence, given two nodes u^r and v^p, one can test whether edge (u^r, v^p) is in E^∞ in time polynomial in the size of H^∞, u^r and v^p. It follows that PETD is in PSPACE, in fact PETD can be solved in non-deterministic linear space and thus in PSPACE by Savitch's theorem as the following algorithm shows:

```
$u^r := x^s$
while  $u^r \neq y^t$
do
        non-deterministically pick a new node $v^p$ in $H^\infty$
        verify $(u^r, v^p) \in E^\infty$
        $u^r := v^p$
od
return(YES)
```

We will see that path existence in td-graphs is actually PSPACE-complete even for very simple status functions. The remainder of this section is devoted to a proof of the following theorem.

Theorem (2.1). *The Path Existence Problem for time-dependent graphs is* PSPACE-*complete.*

Proof. We have already shown that PETD is in PSPACE. To prove PSPACE-hardness we will give a polynomial time transformation from linear bounded automaton (LBA) acceptance to PETD. The basic idea is to construct a td-graph H_n^∞, for all $n \geq 1$, whose vertices v^t can be construed as an instantaneous description of a LBA M during a computation on some input of length n: the current tape inscription is coded by time t and vertex v in H_n codes the current state and head position. H_n consists essentially of n copies of a graph H that represents the finite state control of M, interconnected in a suitable fashion. The whole computation of M then corresponds to a path in H_n^∞ starting at a node that represents the initial configuration with input x.

So let $M=\langle Q, q_I, q_Y, q_N, \delta\rangle$ be a deterministic linear bounded automaton with tape alphabeth $\Sigma=\{0,1\}$; here q_I is the initial state, q_Y and q_N are the accepting and rejecting final states respectively and δ is the transition function. Our notion of Turing machine for this proof will be slightly non-standard; however, it is easy to see that any Turing machine in the sense of a standard definition can be converted into one of our machines (in fact the conversion could be effected by a polynomial time transducer). For the rest of the argument we will assume that M has the following properties:

1) The set of states $Q-\{q_Y, q_N\}$ is partitioned into five classes

QT:	the set of pure transition states
QR:	the set of read states
QW_0:	the set of "write-a-zero" states
QW_1:	the set of "write-a-one" states
QM_L:	the set of "move-head-to-the-left" states
QM_R:	the set of "move-head-to-the-right" states.

In a single step M can either perform a pure transition, a read/write operation or move the tape head but not two at a time. Further we insist that M accepts or rejects only with its head positioned at cell number one; the machine then goes into an infinite loop and stays in the same state q_Y or q_N forever.

2) The only time M uses information on the tape is during a read transition. Therefore the transition function δ can be assumed to have the following format: $\delta=\delta_1\cup\delta_2$ where $\delta_1: QR\times\Sigma\to Q$ and $\delta_2: Q-QR-\{q_Y, q_N\}\to Q$. To simplify notation let $q_i:=\delta_i(q,i)$ for every read state q and $i\in\Sigma$. Let $Q':=\{q_i \mid q\in QR\wedge i\in\Sigma\}$ be the target states of all read transitions. It is safe to assume that for $q\neq p\in QR$ we have $q_i\neq p_j$ for $i,j\in\Sigma$.

3) A write state shall occur only if the currently scanned tape cell contains a symbol different from the desired one. This can be insured by a read transition preceding the write transition (here $p\in QW_1$):

$$\begin{tikzcd}
q_0 \arrow[r] & p \arrow[d, "\text{write 1}"] \\
q \arrow[u] \arrow[r] & q_1
\end{tikzcd}$$

This completes the description of M.

Now suppose $M=\langle Q, q_I, q_Y, q_N, \delta\rangle$ is a LBA as specified. A computation of M on an input $x\in\Sigma^*$ of length n will be simulated by a path in the following td-graph $H_n^\infty=\langle H_n, \Delta_n, \sigma_n\rangle$:

(1) First define the transition diagram of M to be $H:=\langle Q, E\rangle$ where

$$E:=\{(q,\delta_2(q)) \mid q\notin QR\cup\{q_Y,q_N\}\}\cup\{(q,q_0),(q,q_1) \mid q\in QR\}\cup\{(q,q) \mid q\in\{q_Y,q_N\}\}.$$

Introduce a labeling $\lambda: E\to\{-1,0,1\}$ as follows:

$$\lambda(q,p):=0 \quad \text{for } q\notin QM_R\cup QM_L$$
$$\lambda(q,p):=+1 \quad \text{for } q\in QM_R \quad \text{and} \quad \lambda(q,p):=1 \quad \text{for } g\in QM_R.$$

Now let H_n be the product graph $\langle V_n, E_n\rangle$ with vertex set $V_n := Q\times[n]$ and edge set

$$E_n := \{((p,i),(q,j)) \mid (p,q)\in E \wedge i,j\in[n] \wedge j-i=\lambda(p,q)\}.$$

Thus H_n consists essentially of n disjoint copies of H; however, edges starting at a "move-right"/"move-left" vertex q in the i-th copy end in the appropriate vertex p in copy number $(i+1)/(i-1)$. A vertex in H_n thus naturally corresponds to a state p of M together with a head-position i, $1\leqq i\leqq n$.

(2) Now suppose $e=((p,i),(q,j))$ is an edge in H_n. Our objective is to code tape inscriptions by time. More explicitly, for $x\in\Sigma^n$ let val(x) be the natural number whose binary expansion is x; $0\leqq \mathrm{val}(x)<2^n$. Then tape inscription x can be coded by all moments t such that $t\equiv\mathrm{val}(x) \pmod{2^n}$. Thus edge e should have a delay of 2^n unless p is a "write-a-zero" or "write-a-one" state. Writing a 1 into the i-th tape cell for example can be accomplished by assigning a delay of 2^{i-1} (recall our proviso that we only write a 1 if the cell currently contains a 0, i.e. if the i-th digit in the binary expansion of t is 0). Hence Δ: $E_n\to N^+$ is defined as follows.

$$\underset{n}{\Delta}(e)=\begin{cases} 2^n & p\notin QW_0\cup QW_1\cup \in q_Y, q_N\} \\ 2^{i-1} & p\in QW_1 \\ 2^n-2^{i-1} & p\in QW_0 \\ 1 & p=q\in\{q_Y, q_N\}. \end{cases}$$

(3) It remains to define the status function σ_n: $V_n\times N\to\{0,1\}$. σ_n is used to model read operations of M: suppose time t codes some inscription with a 0 in the i-th cell and q is read state. Then q_1 must be blocked at time $t+2^n$, for otherwise a path could go from $(q,i)^t$ to $(q_1,i)^{t+2^n}$. For all other vertices (q,i) with $q\notin Q'$ σ_n is constant 1. Here is the full definition of σ_n:

$$\sigma_n((q,i),t)=\begin{cases} 1 & q\notin Q' \\ 1 & t \textbf{ mod } 2^i<2^{i-1} \\ 0 & 2^{i-1}\leqq t \textbf{ mod } 2^i \end{cases} \quad q=p_0 \text{ for some } p\in QR$$
$$\qquad\qquad\quad \begin{cases} 0 & t \textbf{ mod } 2^i<2^{i-1} \\ 1 & 2^{i-1}\leqq t \textbf{ mod } 2^i \end{cases} \quad q=p_1 \text{ for some } p\in QR$$

To see that this is well-defined recall our convention that q_i must be different from q'_j whenever $q\neq q'$, $q, q'\in QR$. Also note that H_n^∞ has period $\pi=2^n$.

This completes the description of H_n^∞.

An instantaneous description (ID) of M is a triple $C=\langle q,i,x\rangle\in Q\times[n]\times\Sigma^n$. As mentioned above, we use time modulo $\pi=2^n$ to encode tape inscriptions. ID $C=\langle q,i,x\rangle$ thus is represented by any vertex $(q,i)^t$ in H_n^∞ with $t\equiv\mathrm{val}(x)$ (mod 2^n). For two IDs C_1 and C_2 let $C_1\vdash^k C_2$ denote the fact that machine M moves from ID C_1 to ID C_2 in exactly k steps. Now define

$$\mathrm{next}(x,t) := \min(\tau>t \mid \tau\equiv\mathrm{val}(x) \pmod{2^n})$$

where $x\in\Sigma^n$ is a tape inscription and any time $t\in N$. Thus next(x,t) is the next moment in time after t that codes inscription x. Clearly next$(x,t)\leqq t+2^n$. It follows from our definition of H_n^∞ that

$\langle q, i, x\rangle \vdash^1 \langle p, j, y\rangle$ iff
$((q, i)^t, (p, j)^{\text{next}(y,t)})$ is an edge in H_n^∞ for all $t \in N$ such that $t \equiv \text{val}(x) \pmod{2^n}$.

Now consider the source node $(q_I, 1)^{\text{val}(x)}$ in H_n^∞ and the target vertex $(q_Y, 1)$ in H. Clearly

M accepts input $x \in \Sigma^n$
iff $\langle q_I, 1, x\rangle \vdash^N \langle q_Y, 1, y\rangle$ for some $y \in \Sigma^n$, $N \geqq 0$
iff there exists a path in H_n^∞ from $(q_I, 1)^{\text{val}(x)}$ to $(q_Y, 1)^t$ for some $t \geqq \text{val}(x)$
iff $H_n^\infty, (q_Y, 1)^{\text{val}(x)}, (q_I, 1)$ is a YES-instance of PETD.

Note that for any input x of length n the corresponding td-graph H_n^∞ and target vertex are the same, only the source node varies according to x. Instance H_n^∞, $(q_I, 1)^{\text{val}(x)}$, $(q_Y, 1)$ has size $O(|Q|^2 n^3)$ and can thus be constructed from the LBA M and input x in polynomial time. Therefore PETD is PSPACE-hard with respect to polynomial time reductions and we are through. □

Remark. (1) The proof of the last theorem can easily be modified to establish PSPACE-completeness of the node-to-node version of the path existence problem: decide whether there is a path in H^∞ from node v^s to node u^t. Let H_n^∞ be defined as above and set $d := [\lg |Q|] + 2$. Then there are $|Q| \cdot n \cdot 2^n \leqq 2^{dn}$ different ID's of length n of M. Hence M has time complexity $O(2^{dn})$. Now consider the computation C_0, C_1, C_N, of M on input $x \in \Sigma^n$, say $C_i = \langle q_i, j_i, x_i\rangle$, $i \leqq N \leqq 2^{dn}$. To code the tape inscriptions $x = x_0, x_1, \ldots, x_N$ that occur during the computation set

$$t_0 := \text{val}(x) \quad \text{and} \quad t_{i+1} := \text{next}(x_{i+1}, t) \qquad \text{for } i < N.$$

Clearly $t_N < 2^n \cdot 2^{dn} + t_0 < 2^n \cdot 2^{dn} + 2^n =: f(n)$.
Then M accepts input $x \in \Sigma^n$
iff there exists a path in H_n^∞ from $(q_I, 1)^{\text{val}(x)}$ to $(q_Y, 1)^{f(n)}$.

This shows that the node-to-node version of the path existence problem is PSPACE-hard. In fact, the same argument also establishes PSPACE-hardness of the node-to-vertex version of the problem with deadline: pick source node $(q_I, 1)^{\text{val}(x)}$, target vertex $(q_Y, 1)$ and deadline $f(n)$. This is indeed a polynomial time reduction as $f(n)$ can be specified in polynomially many bits.

(2) A very similar construction can be used to simulate a non-deterministic LBA and indeed any Turing machine with a polynomial space bound $S_M(x) \leqq p(|x|)$. The underlying graph H_n then has vertices $Q \times [p(n)]$ and the delays are of the form 2^i for $i \leqq p(n)$. Contrary to the deterministic case H_n^∞ here contains vertices of out-degree larger than one and the (directed) sub-tree of H_n^∞ with root $(q_I, 1)^{\text{val}(x)}$ corresponds to the tree of all possible computations of M on input x.

3. The Simulation

This section is devoted to computational hardness results. Our first objective is to show that one can express the path existence problem for periodic time-dependent graphs introduced in Sect. 2 in terms of a motion planning problem with time-dependent obstacles, thus proving that motion planning is PSPACE-hard. The simulation presented here is in two-dimensional space and uses only rectangular obstacles that are restricted to translational movement. Suppose $H^\infty = \langle H, \Delta, \sigma \rangle$ is some td-graph. The vertices of the static graph H can be represented by rectangular boxes in the plane that are connected via channels corresponding to the edges of H. Body B moves from box to box through these channels tracing a path in H^∞. The argument hinges on the fact that time dependent obstacles moving inside the channels can force B to exit a channel after a certain fixed number of times steps upon first entering it.

Theorem (3.1). *Motion planning in the presence of time-dependent obstacles in two dimensions is* PSPACE-*hard even if the obstacles and the moving body are rectangular and restricted to translational movement.*

Proof: Let $H^\infty = \langle H, \Delta, \sigma \rangle$ be a td-graph with, say, period π. we will show how to translate the reachability problem for H^∞ into a motion planning problem. The obstacle **A** to be described below depends only on H^∞. The reader will find it useful to consult Figs. 3 through 5. We begin by describing the stationary part of **A**. Let n be the number of vertices in H. A vertex v of H with degree d_v will be represented by a rectangular box called box$_v$ that has d_v-many gaps of unit width at the top. The n boxes are lined up along the x-axis and positioned parallel to the axes; the distance between two boxes is n. The width of box$_v$ is 1, its length $2 \cdot d_v - 1$. The gaps of box$_v$ lead into d_v-many

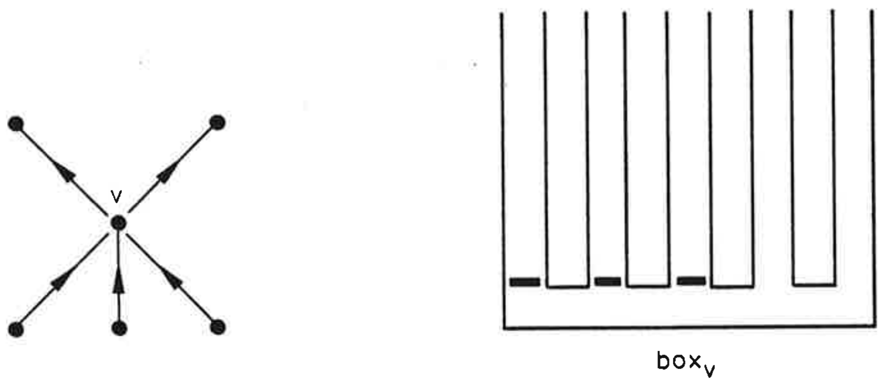


Fig. 3. A vertex in H and the corresponding box

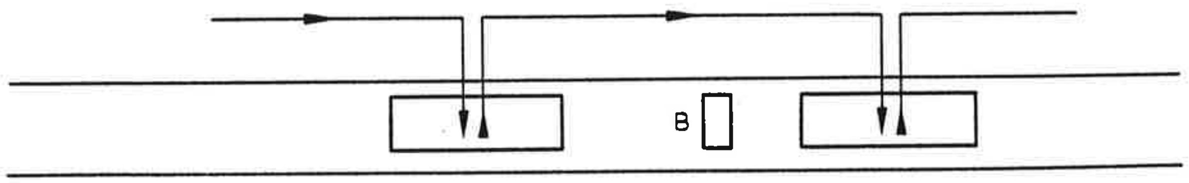


Fig. 4. Movement of a fast obstacle and a safe position for B

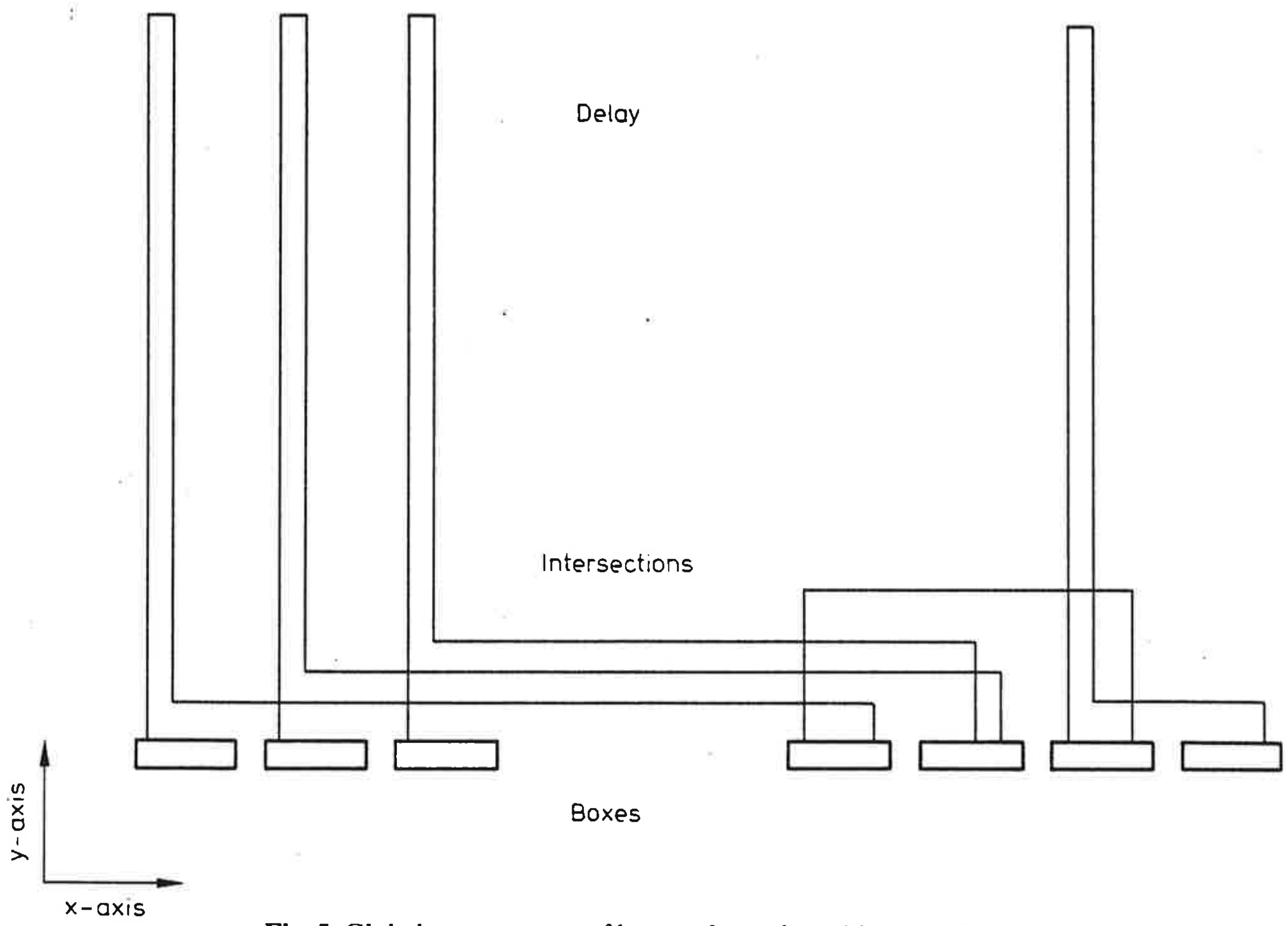


Fig. 5. Global arrangement of boxes, channels and intersections

channels C_e corresponding to edges e of H with source or target v. All channels have width 1 and run parallel to the axes. Boxes and channels are stationary, i.e. they never change position or shape. The horizontal extension of the boxes and channels is thus $O(n^2)$. Body B is a square of length slightly less than 1 and is confirmed to move entirely within the boxes and channels.

One should think of time as being partitioned into slices $S_0, S_1, \ldots, S_t, \ldots$ of equal length θ; thus $S_t = [t\theta, (t+1)\theta]$ where $t \in N$. During time slice S_t box$_v$ represents the node v^t in H^∞. At every gap of a box there is a time-dependent obstacle that appears and disappears once during each time slice, thus closing and opening the gap. The gaps together with their time-dependent obstacles will be called gates. The gates in box$_v$ corresponding to in-edges of v are all closed whenever gates corresponding to out-edges of v are open and vice versa. If vertex $\sigma(v, t) = 0$ then an additional obstacle will block box$_v$ during time slice S_t so that any collision free path has to avoid box$_v$ during the whole slice S_t. Furthermore, at the end of each slice, the whole box is blocked by an obstacle. The interaction of gates and blocking obstacles is intended to force B during each time slice to either enter, pass through and leave a box or to move a little bit further down a channel. As every box has length less than $2n$ we may choose $c = 3n$ to guarantee that B can pass through a box during one time slice. Movement within the channels is forced: if B enters channel C_e during slice S_t it will have to exit (at the opposite end) exactly during slice $S_{t+\delta(e)}$. Here $\delta(e)$ is the delay caused by time dependent obstacles moving inside channel C_e.

There are three major difficulties with this approach:

- First one cannot always choose $\delta(e)=\Delta(e)$: for some edges $e=(u,v)$ in H the delay $\Delta(e)$ may be less than the geometric distance between box_u and box_v. To handle this situation set $\delta(e):=\Delta(e)+\pi\cdot 4n^2$ where π is the period of H^∞. Note that as far as PETD is concerned one may alter the delay $\Delta(e)$ of an edge e by adding multiples of π without affecting the answer. To be more precise, let $H^\infty=\langle H, \Delta, \sigma\rangle$ be a td-graph with period π. Define a new delay function Δ' by $\Delta':=\Delta(e)+k_e\cdot\pi$ where k_e is an arbitrary natural number. Then H^∞, v^s and u is a Yes-instance of PETD iff $\langle H, \Delta', \sigma\rangle$, v^s and u is a Yes-instance of PETD. The geometric distance between box_u and box_v is strictly less than $3n^2$, thus our choice of $\delta(e)$ makes sure that B can move from box_u to box_v in time $\delta(e)$.
- Second let $\Delta_0:=\max(\Delta(e)\,|\,e\in E)\geqq 1$ denote the maximal delay associated with any edge in H. Δ_0 is in general exponential in the size of instance H^∞. Thus one may have to achieve an exponential delay Δ_0 in some channel C_e using only polynomially many atomic obstacles. This can be done by moving $O(1)$ obstacles so quickly that from the point of view of body B (which cannot move faster than c), it appears that there are indeed exponentially many obstacles moving down the channel slowly. See Fig. 4 for the movement of a fast obstacle. The length of the channels as well as the speed of the fast obstacles is $O(\Delta_0+\pi\cdot 4n^2)$ and can thus be specified in polynomially many bits.
- Lastly the channels will in general cross each other, even if H happens to be planar. In order to make sure that B cannot skip from one channel to another there will be four gates at every intersection of two channels, allowing B to move along either channel but not to skip from one to the other. The gates will be open in direction of the x-axis during the first half of every time slice and in direction of the y-axis during the second half. Intersections occur exclusively in the lower part of the arrangement, see Fig. 5 for the global layout of boxes, channels and intersections. Note that this last difficulty does not exist in three dimensions where one could avoid intersections altogether.

Now suppose H^∞, v^s and u is an instance of the node-to-vertex reachability problem for td-graphs. Let obstacle **A** and body B be defined as above. If B is positioned inside box_v at time s then there exists a collision-free path leading to box_u at some time $t\geqq s$ iff there is a path in H^∞ from v^s to $v^{t'}$ for some t', $s\leqq t'\leqq t$. The instance of the reachability problem can clearly be computed in time polynomial in the size of H^∞, v^s and u. With Theorem 2.1 this shows PSPACE-hardness of the motion planning problem with periodic obstacles. □

Remarks. (1) Note that the non-stationary obstacles all can be assumed to move parallel to the axes. Similarly body B could be required to move only parallel to the axes. Hence this restricted version of the motion problem is also PSPACE-hard.

(2) The obstacles in the last proof fail to have disjoint trajectories – and are therefore not readily identified with moving physical objects. However, this can easily be amended: by making the channels and boxes wider one can have all the non-stationary obstacles move entirely within them.

(3) There are various alternatives to the given constructions. One could e.g. code tape inscriptions and head positions by time. Or one could use exponentially many obstacles that all move at speed no more than c and that have a uniform, polynomial description. However, we feel that the argument as given is the most natural and straightforward one.

(4) A direct simulation of LBAs using rotating obstacles in three-dimensional space was described by Reif and Sharir in [RS].

Dimension One: The Street Crossing Problem

We now turn to the one-dimensional situation: B has only one degree of freedom. Motion planning now resembles the problem of crossing by busy street. Say B has to be moved from position $(0, 0)$ to (r, t). Then the interval $[0, r]$ may be construed as a street of width r and B wishes to cross as vehicles pass by. Here B is of course not allowed to jaywalk. Call a c-path monotone if its parametrization is non-decreasing as a function of real numbers, i.e. $P\colon \mathbf{R} \to [0, r]$ such that for all $t_0 \leqq t_1 : P(t_0) \leqq P(t_1)$. We will show that the street crossing problem is NP-hard even if B is required to move along a monotone c-path.

Theorem (3.2). *Motion planning in the presence of time-dependent obstacles in one dimension is NP-hard, even if the obstacles are required to have disjoint trajectories and the movement of B is restricted to be monotone.*

Proof. We will embed 3SAT into the one-dimensional reachability problem. To this end let $U = \{u_1, \ldots, u_n\}$ be a set of Boolean variables and $\Phi = \Phi_1 \wedge \ldots \wedge \Phi_m$ be a boolean formula in 3-conjunctive normal form over U. Thus the clauses Φ_i have the form $\Phi_i = z_{i,1} \vee z_{i,2} \vee z_{i,3}$ where the $z_{i,j}$ are literals over U. The crucial idea is to code truth asignments by time: time $t \in N$ will be interpreted as a truth assignment
$\alpha_t\colon U \to \{\text{true, false}\}$ in the following fashion: let $b_{n-1} \ldots b_1 b_0$ be the binary expansion of t modulo 2^n and set

$$\alpha_t(u_i) := \begin{cases} \text{true} & \text{if } b_{i-1} = 1 \\ \text{false} & \text{if } b_{i-1} = 0 \end{cases}$$

for $i = 1, 2, \ldots, n$.

As in the proof of Theorem 3.1 time is partitioned into slices S_t of, say, length 10. It is convenient to set the top speed c of B equal to 1. We will define a compound obstacle $\mathbf{A}$ depending only on Φ in such a way that B is able to reach the other side iff B starts to move at time t – i.e. at the beginning of slice S_t – such that α_t satisfies all the clauses in Φ. To achieve this, our street will have $m+2$ lanes $L_0, \ldots, L_{m+1}$; each lane except the first and the last consists of five tracks of width 1 each. Lane L_0 and L_{m+1} have only one track of width 1. B is initially positioned in lane L_0 and has to reach lane L_{m+1}. There are no obstacles whatsoever moving on these two lanes, so B

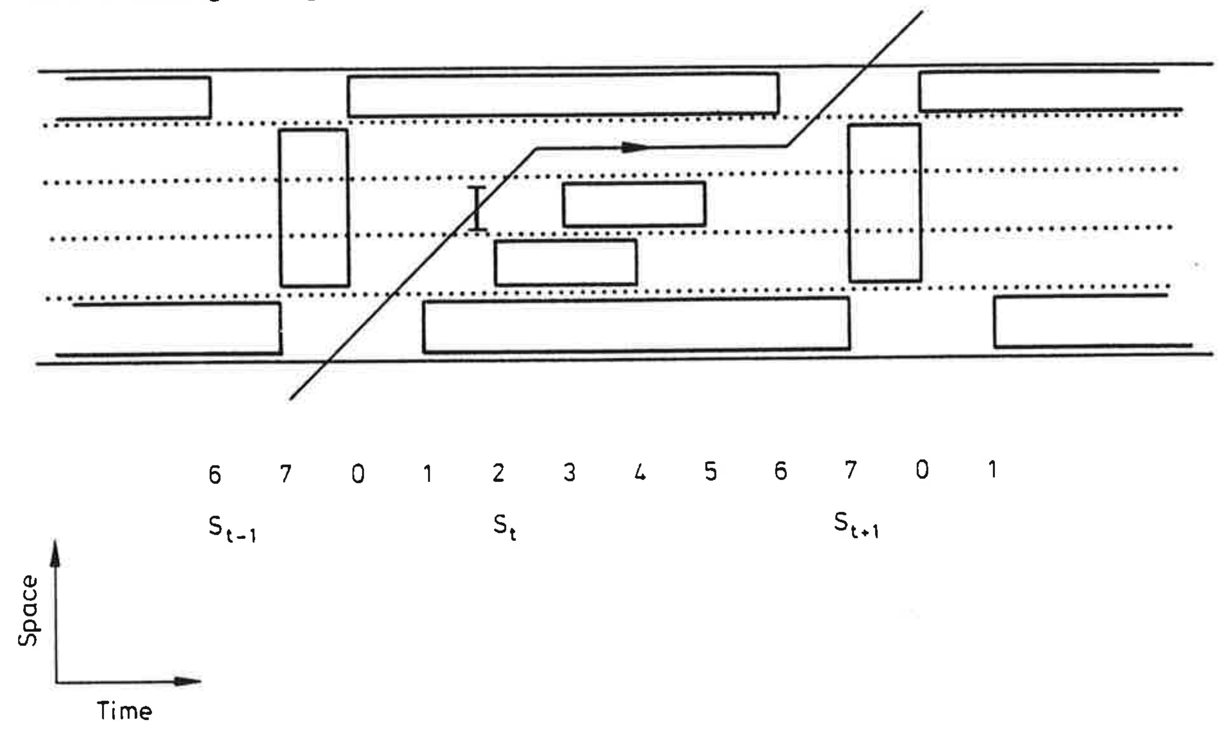


Fig. 6. Movement of obstacles in the 5 tracks of one lane

can stay motionless in lane L_0 for an indefinite amount of time. However, once B has left lane L_0 at time τ and entered L_1 it will be forced to cross one lane during one time slice for the next m time slices until it reaches lane L_{m+1}. In lane L_{m+1} B can again stay motionless forever. Time τ will be referred to as the departure time. Lanes L_i, $i=1, \ldots, m$, will test whether or not α_τ satisfies clause Φ_i: B can successfully cross lane L_i iff α_τ satisfies Φ_i. Therefore B will reach lane L_{m+1} iff α_τ satisfies Φ.

We will now describe the moving obstacles in lane L_i, $i=1, \ldots, m$. Every time slice S_t consists of eight sub-slices $S_{t,0}$, $S_{t,1}$, $\ldots$, $S_{t,7}$ of equal length 5/4. Track 1 has an obstacle of width 1 that is present during sub-slices $S_{t,1}$ through $S_{t,6}$ for all $t \in N$. In track 5 a similar obstacle is present during sub-slices $S_{t,0}$ through $S_{t,5}$, again for all $t \in N$. An auxiliary obstacle of with 3 blocks tracks 2, 3 and 4 during $S_{t,7}$. The satisfaction test is achieved by three obstacles of width 1 each that occur during sub-slice $S_{t,3}$ in tracks 2, 3 and 4. These truth testing obstacles appear periodically and depend on the literals in clause Φ_i. They will appear in lane L_i, track r during slice S_t iff $\alpha_\tau = \alpha_{t-i+1}$ fails to satisfy the corresponding literal $z_{i,r-1}$ of clause Φ_i. If α_{t-i+1} fails to satisfy even a single one of the literals then all three obstacles will be present and form a road block that B cannot pass. In any other case there will be a gap and B will be able to move on to the next lane. To be more precise, in lane L_i, track r ($i \in [m]$, $r \in \{2, 3, 4\}$) during slice S_t an obstacle is present during all sub-slices $S_{t,j}$ such that

$$r \leqq j \leqq r+1 \quad \text{and} \quad \alpha_{t-i+1}(z_{i,r-1}) = \text{false}.$$

Otherwise the obstacles are absent. A thus consists of $6m$ atomic obstacles, six per lane, whose trajectories are clearly disjoint. Figure 6 shows the atomic obstacles in one lane during one time slice.

Let body B be a line segment of length 0.9. Lastly define the source position $p := (0.5, 0)$ and the target location $x' := m + 1.5$. This completes the description of an instance A, B, p, x' of the reachability problem.

For any truth assignment α there is a departure time $\tau < 2^n$ such that $\alpha_\tau = \alpha$. Hence, if Φ is satisfiable at all, lane L_{m+1} can be reached (in fact at some time $\tau' \leqq 2^n - 1 + m + 1 = 2^n + m$). Hence A, B, p, x' is a Yes-instance of our Position-to-Location motion planning problem iff Φ is satisfiable.

Given a boolean formula Φ in 3-conjunctive normal form the instance A, B, p, x' described above can clearly be constructed in time polynomial in the size of Φ. Therefore the Position-to-Location Reachability Problem is NP-hard. □

Is quite straigthforward to modify the last argument to show that the Position-to-Position Reachability Problem and the Position-to-Location Reachability Problem with Deadline are also NP-hard in the one-dimensional situation; they remain NP-hard even if B is required to move along a monotone c-path.

4. Efficient Algorithms for $D = 1, 2$

In this section we will describe algorithmic solutions for certain special cases of the dynamic motion planning problem in low dimensions $D = 1, 2$. Throughout this section the atomic obstacles will be assumed to be convex, compact polytopes in space-time. This includes in particular the case of convex polyhedral objects moving at constant speed without rotation. B will be assumed to be point-like throughout this section. As mentioned earlier one can use the technique of inflating obstacles from [LPW] to reduce the case where B is of polyhedral shape and restricted to translational movement to this situation. Recall that for the running time analysis of our algorithms we use a model of computation that allows for real number as basic objects and provides the necessary algebraic operations such as extraction of square roots, see e.g. [PS].

The One-Dimensional Case

The perhaps most basic approach uses shortest path. Here shortest path is understood as shortest in the sense of the standard Euclidian metric in space-time rather than in space alone. It is easy to see that the shortest c-path – if it exists at all – has to be a polygonal line whose vertices are corners of A. The problem is thus very similar to that of finding the shortest path between to points in two-dimensional Euclidian space avoiding polygonal obstacles. This problem has been studied for example in [LPW, SS, GH] and can be solved by means of the so called visibility graph whose vertices are the corners of A and whose edges are those straight line segments that do not pass through the interior of A. The visibility graph can be constructed in time $O(n^2)$ where n is the number of edges of A (see [GH]); a shortest path can then be found by, say, Dijkstra's algorithm again in $O(n^2)$ steps.

Now define the *c-visibility graph* of **A** to be the directed subgraph of the full visibility graph of obstacle A whose edges correspond to c-paths. By [GH] the c-visibility graph can be constructed in $O(n^2)$ steps. Thus the position-to-position reachability problem can be solved in $O(n^2)$ steps where n is the number of edges of the obstacle. Hence we have the following proposition.

Proposition (4.1). *In one dimension the Position-to-Position Reachability Problem can be solved in time $O(n^2)$ where n is the number of edges of the obstacle. In fact it is possible to find the shortest admissible c-path in $O(n^2)$ steps.*

We will now describe an efficient way to compute the c-hull (and thus solve the escape problem). The algorithm is based on the description of the c-hull afforded by Theorem 1.4 and Lemma 1.5 and uses a plane sweep technique. See [PS] for background information on this algorithmic paradigm. The construction given in Theorem 4.2 can easily be modified to provide solutions to the various reachability problems.

Theorem (4.2). *There is a time $O((n+s)\lg n)$ algorithm to compute the c-hull of a compound compact poyhedral obstacle A where n is the total number of edges of the atomic obstacles of A and s is the number of intersections between these edges.*

In particular, if all atomic obstacles have disjoint trajectories then the c-hull can be computed in time $O(n \lg n)$.

Proof. The core of the algorithm is a sweep procedure that finds the connected components of the c-hull of A using a x-structure and a t-structure. The sweep line $H=H_t$ is parallel to the x-axis and moves from the future back into the past. Let $C:=c\text{-hull}(\mathbf{A})$. We wish to compute the intersection $H_t \cap C$ between the sweep line and the c-hull which is a collection of disjoint intervals $I_1, \ldots, I_k$, called active intervals. During the livespan of an active interval its endpoints move along the boundary of a connected component of C. More precisely, according to Lemma 1.3, an active interval I evolves during the sweep as follows: I is created when a new edge outside of the currently active intervals is encountered. Then its endpoints hi_I and lo_I either follow an edge of **A** or they move inward at speed c. Eventually they meet, i.e. the interval shrinks to a point and disappears. In the first case endpoint $q \in \{hi_I, lo_I\}$ is associated with an edge $l(q)$ of **A**. In the second case hi_I and lo_I must be escape-points and can thus be associated with two rays $r^{\downarrow}(hi_I)$ and $r^{\uparrow}(hi_I)$ that start at the endpoints, have inclination γ and point backwards in time towards the center of I. Let us call these rays and edges of **A** that intersect H_t active lines (at time t). Active lines are stored in the x-structure, sorted according to the x-coordinate of the point of intersection between H_t and the line. Every active line has a status: it is external if it currently makes up part of the boundary of C and internal otherwise. Rays are always external. Clearly a change in the x-structure can occur at time t only if an intersection occurs, a new edge is encountered or a previously active edge ends at time t. Note that an intersection can occur only between two adjacent lines in the x-structure. We will refer to such a time t as a critical moment.

The t-structure is initialized to contain $T(p)$ and $T(q)$ for every edge $[p, q]$ of **A** sorted in non-increasing order. During the sweep additional entries will be made for those times t at which active lines intersect. At a critical moment the following changes can occur in the x-structure:

(1) A new edge is encountered and added to the x-structure. New edges are internal or external depending on their location relative to the currently active lines.

(2) A previously active edge becomes inactive and is delected from the x-structure.

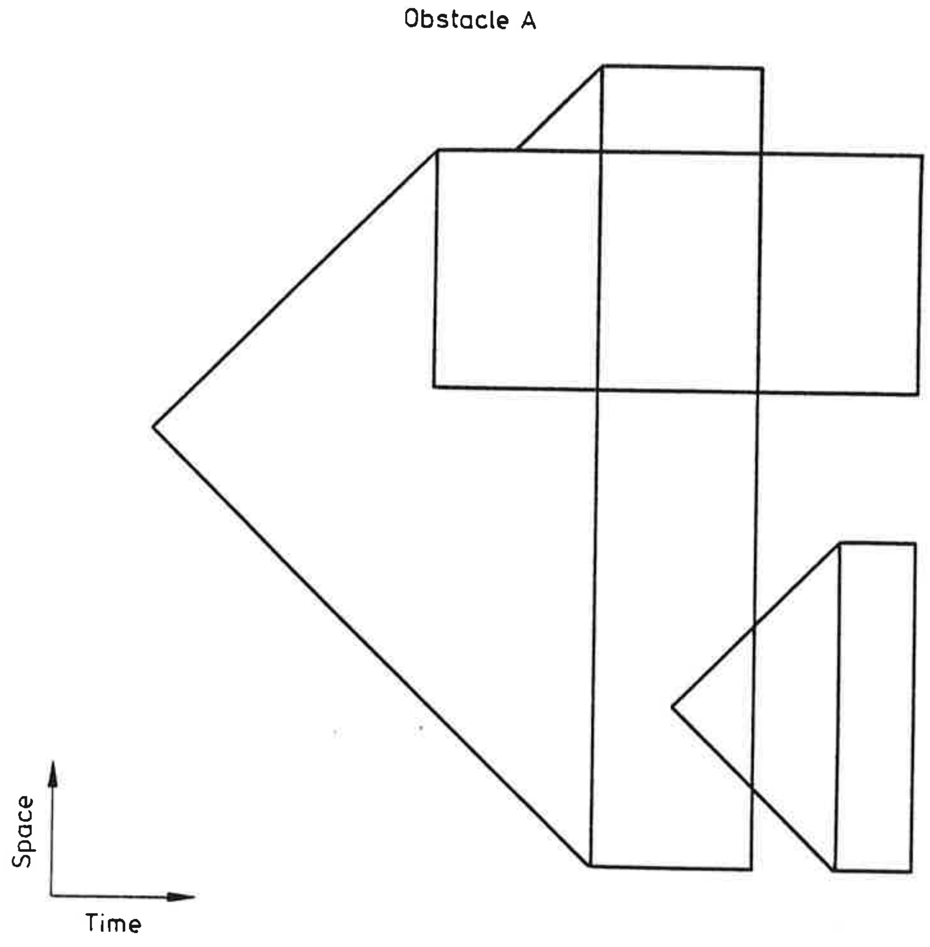
(3) Two active lines intersect.

Note that these events are not mutually exclusive. For example an edge parallel to the x-axis will become active and inactive at the same critical moment. In any case, it is quite straightforward to deal with situation (1) and (2). As far as intersections of active lines are concerned a ray is always truncated as soon as it intersects any active line. If that other active line is also a ray we are necessarily in a situation where $r^{\downarrow}(hi_I)$ intersects $r^{\uparrow}(lo_I)$. Whence interval I is deleted at that time. If that other active line is an edge it becomes external. If both intersecting lines are edges they are switched in the x-structure. If they are both external there are two possible scenarios: they belong to the same active interval which is accordingly deleted or they belong to different active intervals which are accordingly merged. If exactly one of them was previously external they switch status. If both were previously internal they remain so. In short the algorithm looks as follows:

```
initialize t-structure and x-structure
while
          t-structure is not empty
do
          t := largest entry in the t-structure
          delete t
          (possibly) change status of active lines
          recalculate critical moments
od
```

Figure 7 illustrates the stages in the execution of the algorithm on a simple obstacle $\mathbf{A} = A_1, A_2, A_3$ where $A_1 = [6, 9] \times [-2, 4]$, $A_2 = [0, 10] \times [0, 2]$, $A_3 = [0, 4] \times [3, 4]$. The table shows the t-structure and the active intervals. The endpoints of active intervals that are associated with rays are indicated by an arrow $\uparrow$ (for a ray associated with $r^{\uparrow}(lo_I)$) or $\downarrow$ (for a ray associated with $r^{\downarrow}(hi_I)$). All other endpoints are associated with edges of **A**. Entries in the t-structure made during the execution of the algorithm are underlined. The speed c is assumed to be 1. The algorithm terminates at time -5.5 when the t-structure becomes empty.

A data structure that supports the necessary operations for the x-structure such as insertion, deletion, find and interchange in logarithmic time is for example a dictionary. The status of an active line is indicated by an additional bit. The t-structure can be implemented as a priority queue. To keep track of the



Time t	t-structure	Active intervals	
initialization	4, 3, 2, 0, -2	empty	
4	3, 2, 0, -2	$I_1=(0, 4)$	$I_2=(6, 9)$
3	2, 1, 0, -2	$I_1=(0\uparrow, 4\downarrow)$	$I_2=(6, 9)$
2	0, -2	$I_1=(0, 10)$	
0	-1, -2	$I_1=(0\uparrow, 10\downarrow)$	
-1	-2	$I_1=(1\uparrow, 9)$	
-2	-5.5	$I_1=(2\uparrow, 9\downarrow)$	
-5.5	empty	empty	

Fig. 7. Computing the c-hull in one dimension using the plane sweep method of Theorem 5.2. Obstacle **A** has the form $\mathbf{A}=A_1, A_2, A_3$ as indicated. The table shows the t-structure and x-structure at various stages in the execution of the sweep. See the text for an explanation

boundary of the connected components of C as described by the external active lines one may use mergeable dequeues.

For the running time analysis observe that the total number of active lines in $O(n)$. Therefore the data structures used inside the while loop all have size $O(n)$ and the basic operations in the loop take $O(\lg n)$ steps. The while loop is executed $O(n+s)$ times where s is the number of intersections of edges of the atomic obstacles. Initializing the t-structure takes $O(n \lg n)$ steps. Hence the total running time of the algorithm is $O((n+s) \lg n)$ steps. □

Remarks. The above algorithm is optimal in the decision tree model: sorting can be linear-time reduced to the computation of the c-hull. Say $r_1, \ldots, r_n$ are pairwise different real numbers. Define $A_i := [-b, +b] \times [r_i, r_i + \delta] \subset \mathbf{R}^2$ for $i = 1, \ldots, n$ where b is chosen large enough to insure that the c-hull of the compound obstacle $\mathbf{A} = A_1, \ldots, A_n$ is connected, say $b := 2 \cdot \tan(\gamma) \cdot (\max r_i - \min r_i)$, and δ is small enough to make sure that the atomic obstacles A_i do not overlap, say $\delta := \min(|r_i - r_j| \mid i \neq j)/2$. It is easy to read off the ordered sequence from the c-hull of A.

Theorem (4.3). *There is a time $O((n+s) \lg n)$ algorithm that solves the Position-to-Position Reachability Problem for a compound compact polyhedral obstacle A where n is the total number of edges of the atomic obstacles of A and s is the number of intersections of these edges.*

If in particular the trajectories of all atomic obstacles are disjoint then the problem can be solved in time $O(n \lg n)$. The same holds for the Position-to-Location Reachability Problem and the Position-to-Position Reachability Problem with Deadline.

Proof. Let $\mathbf{A}$ be a compound obstacle $A_1, \ldots, A_m$, $p_0 = (x_0, t_0)$ the source and $p_1 = (x_1, t_1)$ the target position. The algorithm is very similar to the one given in the last proof: the only difference is that active intervals here represent the intersection of $R(p; \mathbf{A})$ and the sweep line. Therefore active intervals now expand rather than shrink, i.e. their endpoints move outward with speed c or they follow an edge of $\mathbf{A}$. The sweep starts at time t_0 and terminates at t_1. p_1 is reachable from p_0 iff upon termination of the sweep location x_1 is contained in one of the active intervals: for some interval I we must have $lo_I \leqq x_1 \leqq hi_I$.

Now suppose we are given a source position $p_0 = (x_0, t_0)$ and a target location x_1. Set $t_1 := \max(T(p) \mid p \in \mathbf{A}) + 1$. t_1 is initially entered into the t-structure together with the entries described above. Conduct a sweep starting at time t_0 as before and maintain in the x-structure the intersection of $R(p; \mathbf{A})$ and the sweep line. Halt and return YES if for the first time $t \leqq t_1$ one of the active intervals contains x_1 or if t_1 is reached and there is at least one active interval at time t_1. Otherwise return NO. A deadline T is handled by truncating the sweep at time T. □

The Two-Dimensional Situation

We will now briefly describe how to modify the above algorithms in the two-dimensional situation. One major difficulty is that the intersection between the c-hull C and the sweep plane H_t is now a collection of disjoint regions that have no nice linear arrangement as the active intervals in the one-dimensional situation do. Unless the obstacle is convex we now also have to deal with non-planar surface patches of the c-hull even through $\mathbf{A}$ is a polyhedron in space-time. A typical example is provided by $\mathbf{A} := A_1 \cup A_2$ where $A_1 := [0, 1] \times [0, 2] \times [0, 1] \subset \mathbf{R}^3$ and $A_2 := [0, 2] \times [0, 1] \times [0, 1] \subset \mathbf{R}^3$. $\mathbf{A}$ is L-shaped and present for one time unit. Positions $p = (1 - \delta_1, 1 - \delta_2, -\tau)$ for $0 < \delta_1, \delta_2, \tau \ll 1$ on the boundary of the c-hull all have $q_p := (1, 1, 0)$ as escape-point, so the c-hull must have a part of $\text{cone}_-(1, 1, 0)$ as its boundary.

However, the next lemma shows that this is the only possible complication.

Lemma (4.4). *Let A be a polyhedral obstacle, C its c-hull. Then the boundary of C consists of planar portions and segments of cones. Thus for any time t the cross section $\partial C \cap H_t$ consists only of line segments and circular edges.*

The same holds for the set of reachable points $R(p; \mathbf{A})$.

Proof: Again we will only consider the c-hull, the argument for the set of reachable points is quite similar. So let $C := c\text{-hull}(\mathbf{A})$ and consider some position $p=(x, t)$ in $\partial(C \cap H_t) = \partial C \cap H_t$. If p lies on the boundary of $\mathbf{A}$ our claim is immediate from the fact that $\mathbf{A}$ is polyhedral. So assume that $p \notin \partial\mathbf{A}$ and let $q_p \in \partial\mathbf{A}$ be an escape-point for p. It is easy to see that q_p cannot be an interior point of a face of $\mathbf{A}$. So q_p is either an interior point of an edge L of $\mathbf{A}$ or a vertex of A. Define for edges L and vertices q of $\mathbf{A}$ the following surface patches of C:

$$C_L := \{p \in \partial C - \partial\mathbf{A} \mid q_p \text{ is an interior point of } L\} \quad \text{and}$$
$$C_q := \{p \in \partial C - \partial\mathbf{A} \mid q_p = q\}.$$

Then $\partial C - \partial\mathbf{A}$ is the union of finitely many pieces C_L and C_q. Now the line segment $[p, q_p]$ lies on $\text{cone}_-(q_p)$ and we have $[p, q_p] \subset C_L \subset \partial C$. The surface patch C_L can therefore be generated by moving the apex q_p of $\text{cone}_-(q_p)$ along the edge L. So C_L is planar. In the second case observe that C_q is actually a portion of $\text{cone}_-(q_p)$. □

As a corollary to the proof of Lemma 4.4 we obtain a bound $O(n)$ on the number of planar and conic surface patches of the c-hull where n is the total number of edges of the compound obstacle $\mathbf{A}$. Also note that the planar pieces σ that are not part of the boundary of $\mathbf{A}$ all have inclination γ and are tangent to an edge $[p, q]$ of $\mathbf{A}$. We will say that $[p, q]$ supports σ. The cones are all of the form $\text{cone}_-(p)$ for some vertex of $\mathbf{A}$, hence any plane σ of inclination γ is parallel to a tangent plane of $\text{cone}_-(p)$.

Now consider the problem of computing the c-hull C of $\mathbf{A}$. According to Lemma 4.4 ∂C consists only of pieces of surface that can be described by algebraic equations of degree at most two. In particular the boundary of these surface patches is also algebraic and of degree at most two. As a matter of fact there are only three possible types of boundaries:

- Line segments. These occur as edges of $\mathbf{A}$ and more generally at the intersection of two planar surface patches but also at the intersection of $\text{cone}_-(p)$ and the plane σ supported by edge $[p, q]$.
- Parabolas occur at the intersection of a cone $\text{cone}_-(p)$ and the plane σ supported by edge $[q_1, q_2]$ where $p \notin \{q_1, q_2\}$ as $\text{incl}(\sigma) = \gamma$.
- Hyperbolas arise at the intersection of a cone $\text{cone}_-(p)$ and planar surface patches that are part of the boundary of $\mathbf{A}$.

Recall that we are using a real RAM as a model of computation. Therefore we may treat all these geometric objects as primitive and charge only one step for operations such as determining the intersection of a cone and a plane.

Theorem (4.5). *Let $\mathbf{A}$ be a compound compact polyhedral obstacle. There is an algorithm to compute the c-hull of $\mathbf{A}$ that is polynomial in number of edges of $\mathbf{A}$.*

Theorem (4.6). *There is an algorithm that solves the Position-to-Position Reachability Problem for a compound compact polyhedral obstacle with running time polynomial in the number of edges of the obstacle. The same holds for the Position-to-Location Reachability Problem and the Position-to-Location Reachability Problem with Deadline.*

Proof. For the sake of simplicity we will focus on the algorithm for the computation of the c-hull; the decision procedure for reachability is quite similar. The algorithm is based on a space sweep (or rather a space-time sweep) where the sweep plane H_t is parallel to the x, y-plane and moves from the future into the past. The intersection of the sweep plane and the c-hull C of $\mathbf{A}$ is a collection of connected mutually disjoint regions, called active regions, that change dynamically as the sweep plane advances. Let R be an active region; its boundary ∂R is a closed curve on H_t consisting of line segments and circular edges, say $\partial R = l_1, \ldots, l_m$. A line segment l_i will be represented by the plane it lies on (which is either a face of $\mathbf{A}$ or a tangent plane of inclination γ supported by some edge $[p, q]$ of $\mathbf{A}$) and the trajectory of its endpoints (which is a quadratic curve by the preceding discussion). Similarly a circular edge is represented by the cone it lies on (which of the form $\text{cone}_-(p)$ for some vertex p of $\mathbf{A}$) and the trajectory of its endpoints (which is again a quadratic curve). These boundaries are stored in the x, y-structure as are the edges of $\mathbf{A}$ that are currently intersected by H_t. Again they will be referred to as active lines. Critical moments now occur whenever a change in the active lines happens. Again there are three cases: a new edge of $\mathbf{A}$ is encountered and becomes active, a previously active edge becomes inactive or, lastly and most importantly, the boundary of an active region changes. The t-structure is initialized to contain the t-components of all the vertices of $\mathbf{A}$ ordered in non-increasing fashion. As the sweep proceeds additional entries are made for the critical moments of the last type. In short the algorithm can now be described as follows:

```
while
        $t$-structure is not empty
do
        $t$ := largest entry in the $t$-structure
        delete $t$
        update active lines in $x, y$-structure
        update boundaries of active regions
        re-calculate critical moments
od
```

Let n be the total number of all edges of the atomic obstacles A_i of $\mathbf{A}$. For the running time analysis first note that the number of critical moments is linear in the number of edges of the compound polyhedron $\mathbf{A}$ which is $O(n^3)$ as atomic obstacles are not required to have disjoint trajectories. Therefore the while-loop is executed $O(n^3)$ times. The total number of boundary segments of the active regions at any moment during the sweep is $O(n^2)$. Thus one can compute the next critical moment in $O(n^4)$ steps by explicitly determining all the possible interactions between these segments. Similarly updating the active region can be handled in $O(n^4)$ steps. Initialization of the t-structure takes only $O(n \lg n)$ steps, so the whole algorithm runs in time $O(n^7)$. $\square$

Remark. A substantial improvement in the running time of the last algorithm is possible if the obstacle A is convex, compact and polyhedral. According to Corollary 1.6 the c-hull C is then also convex, compact and polyhedral. During the sweep there is only one active region $C \cap H_t$ and its boundary consists of $O(n)$ the segments. Interactions now occur solely between adjacent line segments. As a matter of fact the projection of the edges of the planar surface patches of ∂C that are not part of the boundary of $\mathbf{A}$ into the x, y-plane is a planar graph and even a tree. The c-hull can be constructed in $O(n \lg n)$ steps.

Another special case that allows for a faster solution is when the obstacle is of the form $A = A_0 \times [t_0, t_1]$ where $A_0 \subset \mathbf{R}^2$ is a polygonal region of the plane (but not necessarily convex). The projection of the boundaries of the surface patches of ∂C into the x, y-plane is the generalized Voronoi diagram of A_0: any position p on the boundary of a surface patch of ∂C, $p \notin \mathbf{A}$, has at least two different points p'_1 and p'_2 and p is equidistant to p'_1 and p'_2 in space-time as $T(p'_1) = T(p'_2) = t_0$. As $\mathrm{incl}([p, p'_i]) = \gamma$ for both $i=1$ and $i=2$ equidistance is preserved under the projection. Conversely every point (x, y) of the Voronoi diagram can be associated with a position $p = (x, y, t)$, $t < t_0$, that lies on the boundary of a surface patch of ∂C. The Voronoi diagram of A_0 contains straight-line segments and parabolas and can be constructed in $O(n \lg n)$ time, see [K].

Acknowledgements. We would like to thank the unknown referees for many helpful comments and suggestions.

References

[DS] Dunford, N., Schwartz, J.: Linear operators. Part I: General theory. New York: J. Wiley & Sons, 1964

[GH] Guibas, L., Hershberger, J.: Computing the visibility graph of n line segments in $O(n^2)$ time. Bull. EATCS **26**, 13–19 (1985)

[HJW] Hopcroft, J., Joseph, D., Whitesides, S.: On the movement of robot arms in 2-dimensional bounded regions. SIAM J. Comput. **14**, 315–333 (1985)

[HSS] Hopcroft, J.E., Schwartz, J.T., Sharir, M.: On the complexity of motion planning for multiple independent objects; PSPACE-hardness of the "warehouseman's problem". Int. J. Robot. Res. **3(4)**, 76–88 (1984)

[K] Kirkpatrick, D.G.: Efficient computations of continuous skeletons. Proceedings of the 20th IEEE Symposium on Foundations of Computer Science, 1979. pp. 18–27. IEEE Press, New York

[LPW] Lozano-Perez, T., Wesley, M.: An algorithm for planning collision-free paths among polyhedral obstacles. Commun. ACM, 560–570 (1979)

[PS] Preparata, F.P., Shamos, M.I.: Computational geometry. Berlin Heidelberg New York: Springer 1985

[R] Reif, J.: Complexity of the mover's problem and generalizations. Proceedings of the 20th IEEE Symposium on Foundations of Computer Science, 1979. pp. 421–427. IEEE Press, New York

[RS] Reif, J., Sharir, M.: Motion planning in the presence of moving obstacles. Harvard University, TR-06-85

[SS1] Schwartz, J.T., Sharir, M.: On the piano mover's problem. I. The special case of a rigid polygonal body moving amidst polygonal barriers. Commun. Pure Appl. Math. **36**, 345–398 (1983)

[SS2] Schwartz, J.T., Sharir, M.: On the piano mover's problem. II. General techniques for computing topological properties of real algebraic manifolds. Adv. Appl. Math. **4**, 298–351 (1983)

[SS3] Schwartz, J.T., Sharir, M.: On the piano mover's problem. III. Coordinating the motion of the several independent bodies: the special case of circular bodies moving among polygonal barriers. Int. J. Robot. Res. **2(3)**, 46–75 (1983)

[SY] Spirakis, P., Yap, C.: Strong NP-hardness of moving many disks. Inform. Process. Lett. **19**, 55–59 (1984)

[Y] Yap, C.: Coordinating the motion of several disks. TR-105-84, Courant Institute of Math. Sciences, New York University, February 1985

Received December 12, 1985 / May 6, 1988