Continuous Alternation:
The Complexity of Pursuit in Continuous Domains*

John H. Reif Stephen R. Tate
Computer Science Department
Duke University
Durham, NC 27706

Abstract

Complexity theory has used a game theoretic notion, namely alternation, to great advantage in modeling parallelism and in obtaining lower bounds. The usual definition of alternation requires that transitions be made in discrete steps. The study of differential games is a classic area of optimal control, where there is continuous interaction and alternation between the players. Differential games capture many aspects of control theory and optimal control over continuous domains. In this paper, we define a generalization of the notion of alternation which applies to differential games, and which we call "continuous alternation". This approach allows us to obtain the first known complexity theoretic results for open problems in differential games and optimal control.

We concentrate our investigation on an important class of differential games, which we call polyhedral pursuit games. Pursuit games have application to many fundamental problems in autonomous robot control in the presence of an adversary. For example, this problem occurs in manufacturing environments with a single robot moving among a number of autonomous robots with unknown control programs, as well as in automatic automobile control, and collision control among aircraft and boats with unknown or adversary control.

We show that in a 3-dimensional pursuit game where each player's velocity is bounded (but there is no bound on acceleration), the pursuit game decision problem is hard for exponential time. This lower bound is somewhat surprising due to the sparse nature of the problem: there are only two moving objects (the players), each with only three degrees of freedom. It is also the first provably intractable result for any robotic problem with complete information; previous intractibility results have relied on complexity theoretic assumptions.

Fortunately, we can counter our somewhat pessimistic lower bounds with polynomial time upper bounds for obtaining approximate solutions. In particular, we give polynomial time algorithms that approximately solve a very large class of pursuit games with arbitrarily small error. For any $\epsilon > 0$, this algorithm finds a winning strategy for the evader provided that there is a winning strategy that always stays at least ϵ distance from the pursuer and all obstacles. If the obstacles are described with n bits, then the algorithm runs in time $(n/\epsilon)^{O(1)}$, and applies to several types of pursuit games: either velocity or both acceleration and velocity may be bounded, and the bound may be of either the L_2 or L_∞ norm. Our algorithms also generalize to when the obstacles have constant degree algebraic descriptions, and are allowed to have predictable movement.

*Supported in part by Air Force Contract AFOSR-87-0386, DARPA/ISTO contract N00014-88-K-0458, NASA subcontract 550-63 of prime contract NAS5-30428, and US-Israel Binational NSF Grant 88-00282/2.

1 Introduction

In alternation (as presented by Chandra, Kozen, and Stockmeyer [7]), the state configurations are finite strings, and two players (an existential and universal player), beginning at an initial configuration, alternately make discrete moves chosen from a given finitely described next move relation. The players make these moves with perfect information of the current position; the associated decision problem is to determine whether there is a strategy for the existential player that always reaches a given final accepting configuration. Discrete alternation has proved to be a very useful notion, with applications in complexity theory, game theory, and parallel computing.

We investigate an interesting variant of alternation, which we call continuous alternation. Each configuration is a point in a dense space, say $\Re^d$, and there are again two players: the existential and universal players. The configuration x is partitioned into two parts, where each part is controlled by a distinct player. Each player continuously makes moves satisfying differential constraints of the form $F(t, x, x', x'', ...)$, where F defines a set of semialgebraic inequalities in the derivatives of x and their norms. We also specify distinguished initial and final configurations and their derivatives. Again, both players have perfect information, and the problem is to determine a continuous strategy for the existential player that is successful in reaching the final configuration with a continuous trajectory against any dynamic pursuer. In addition, there may also be a restriction on the time required to reach the final configuration. Interesting examples of continuous alternation games are the differential pursuit games considered in game theory (see, for example, [2]). A typical pursuit game has constraints $F(t, x, x', x'', ...)$, where F specifies the geometry of obstacles to be avoided by players, the shape of the players, as well as a restriction that the players not collide.[1] F can also specify a norm bound on the velocity or acceleration of each player. Such pursuit games are actually just robotic motion planning problems in the presence of an adversary that tries to stop our robot. A short summary of previous work in motion planning is included in section 2.

The complexity of continuous alternation depends very much on the form of the differential constraints. We provide the first upper and lower bounds on the complexity of a class of continuous alternation games. In particular, we consider pursuit games in 3 dimensions, where the obstacle sets are polyhedra with fixed rational position in $\Re^3$, and the L_2 norm of the velocity of each player is bounded. We show that the decision problem for these pursuit games is exponential time hard. The lower bound is quite surprising, since the degree of freedom of the players (the dimension d of the configuration space) is constant. This is the first provable intractability result for a robotics problem with perfect information. In fact, there had previously been no provable intractability

[1] A collision is defined as an intersection of the players that has non-zero volume. In other words, when avoiding collisions the boundaries are allowed to intersect, but no points internal to the players may overlap.

results for any robotic problems with perfect information, even with n degrees of freedom. Note that there are problems with perfect information that have been shown to be NP-hard or even PSPACE-hard; however, to say that a PSPACE-hard problem is provably intractable requires a proof that P $\neq$ PSPACE. While this is considered a reasonable *assumption* by many computer scientists, it remains a major open problem in complexity theory.

We give approximation algorithms for a wide class of pursuit games: the velocity and acceleration have either L_∞ or L_2 norm bounds, the obstacle sets are (possibly moving) polyhedra with fixed rational initial positions, and there are certain "safety" constraints on the closeness that the players can approach obstacles or each other. Our approximation algorithms allow the existential player to find a strategy that reaches the final configuration within an additive ϵ factor (for any $\epsilon > 0$) of the optimal deadline time. In the case of L_2-norm bounds, we have to exceed the dynamics bounds by a multiplicative factor of ϵ. The algorithm generalizes to any type of obstacle (not just polyhedra) that can be tested for collision in $O(\log n)$ space, such as obstacles having constant degree algebraic descriptions.

To emphasize the relationship with discrete alternation, both the lower and upper bounds are proved using the alternating Turing machine as the model of computation.

There are many variations on the pursuit game we have defined, and some minor changes can greatly affect the complexity of the problem. For instance, if each player is a single point, then it can be shown that whenever the pursuer is allowed a velocity bound at least as high as the evader, the game is easily decided by just calculating the minimum distance to the goal for each player.

2 Previous Work in Motion Planning

2.1 Motion Planning for Static Problems with Perfect Information

We first describe the static obstacle motion planning problem, which is to move the robot between two given locations, avoiding contact with fixed obstacles. The problem has been formulated (see for example, Lozano-Perez and Wesley who called it the "FIND-PATH problem" [15], Reif who called it the "furniture mover's problem" [18], and Schwartz and Sharir who called it the "piano mover's problem" [26]) by describing the robot as a linked set of polyhedral bodies, and the obstacles as a set of static polyhedra, fixed in given positions in 3-space. Early work in this area includes: *obstacle growing* techniques [15], and complexity theoretic techniques, such as the first known PSPACE-hardness result for n-linked polyhedral robots [18, 19]. Later work by Schwartz and Sharir developed polynomial time motion planning algorithms for the case of robots with a constant number of degrees of freedom and size n obstacle space [26]. The Schwartz and Sharir algorithm constructs a partitioning of the configuration space by decomposing it into a simplicial complex. Their general algorithm used in part the Collins decomposition developed originally to decide the theory of real closed fields (in fact, to decide the theory of real closed fields with a constant number of variables, there was a previously known polynomial time sequential algorithm due to Collins [9], but this does not give the path information needed to solve these robotic movement planning problems).

2.2 Minimum Distance Path Planning with Perfect Information

In the simplest formulations of minimal distance (static obstacle) path planning, the robot is assumed to be a point, or sphere. We wish to determine the minimal distance path between two points for the robot that avoids a set of n fixed polyhedral obstacles. Interesting techniques have been developed to solve these problems; such techniques might be applied to solve many other robotic problems. For example, Sharir and Schorr give a doubly exponential sequential time algorithm for the 3-D minimum path problem using the theory of real closed fields [27]; later, Reif and Storer apply a formula collapsing trick that (aided by later work of Sharir) reduced the complexity to single exponential sequential time or $n^{O(\log n)}$ parallel time (using a large number of processors) [20]. The work of Canny [3] and Renegar [23] on the existential theory of real closed fields further reduced the parallel time and sequential space complexity to polynomial, but the total work (and sequential time) remains non-polynomial.

It will be difficult to reduce the work bounds of this problem to polynomial, since Canny and Reif proved that the 3-D minimum distance path problem is NP-hard [6]. A key part of that proof uses the fact that we can construct an exponential number of homotopic distinct paths through a set of $O(n)$ convex obstacles. Finally, we note that the 2-D minimum path problem can be solved in polynomial time; the best known algorithm is $O(nk \log n)$ time [20], where k is the number of disconnected polyhedral obstacles. Quadratic bounds are known when $k = n$, and this case has known parallel algorithms. (See also [21] for variants of the 2-D minimum path problem where the number of turns is minimized). In a more practical vein, Papadimitriou gives a polynomial approximation algorithm for finding 3-D minimum cost paths using space discretization [17]; for any $\epsilon > 0$, this algorithm will find a path that is provably within a $(1+\epsilon)$ factor of optimal. The running time for this approximation problem has been improved by Clarkson, as well as giving an $O(\frac{n}{\epsilon} \log n)$ approximation algorithm for two dimensions [8].

2.3 Compliant Motion Control and Frictional Movement Planning

The inclusion of friction complicates movement planning in interesting ways. In this case there is incomplete information about the position of the robot. In fact, in compliant control the lack of knowledge and imprecision of the robot position can grow in time (see also [10]). Canny and

Reif showed that 3-D compliant motion control is nondeterministic exponential time hard in the worst case [6], which was the first provably intractable result in robotics. These lower bound results required incomplete information, whereas the problems addressed in this paper assume that perfect information is available.

2.4 Motion Planning for Autonomous Robots in Dynamic Environments

A more realistic situation for robot systems of the future is an army of autonomous robots operating in an environment that is not “nice”. Due to the existence of other autonomous robots (and possibly other factors) that act as moving obstacles, the environment is not static, but can be changing and dynamic. This complication requires dynamic algorithms for control. With the development of such controls, movement planning may eventually be relegated to a background process of a robot. As very little is currently known about such problems, it remains a fundamental problem to develop dynamic movement planning algorithms; this is the main thrust of our current paper.

2.5 Dynamic Motion Planning with Independently Moving Obstacles

One way that we can generalize the mover's problem is to allow the obstacles to move; we assume that the obstacles are moving with known trajectories. We then must move the robot on a path between two given positions, with bounded velocity while avoiding the moving obstacles. This problem occurs in manufacturing environments with many other autonomous pre-programmed robots, automatic automobile control, and automatic pilots and collision control for aircraft and boats. The first papers with complexity results (i.e., algorithms and lower bounds for motion planning with moving obstacles) were those of Reif and Sharir [20], and Sutner and Maas [29]. Reif and Sharir develop algorithms for the 3-D movement planning problem with translating (but non-rotating) polygonal obstacles, and give PSPACE lower bounds for 3-D movement planning of a disk, among rotating obstacles [20]. This construction uses $\Omega(n)$ rotating obstacles, while the lower bound construction of this paper only needs two moving objects.

2.6 Kinodynamic Motion Planning

In kinodynamic motion planning, the kinematic and dynamic constraints of the robot are taken into account, so driving forces, torques and velocities of the robot are restricted by some maximum allowed norm. The formulation may vary depending on whether L_2 or L_∞ norm is used, and whether the full kinodynamic equations are used. The minimum time problem is to move the robot (say a set of linked polyhedra) between two given positions, with beginning and final velocities also specified. For dimension $d = 1$, the problem can be easily solved [16]. For $d > 1$, this problem is a very difficult one, and it dates back to at least the last century. There has been some considerable previous work done by robotic researchers. For example, Bobrow [1], and later Shin and McKay [28], proved some basic constraints on control paths; for example, at least some joint torque is at maximum (see also [25]). Hollerbach gives a dynamic time scaling technique [13], which allowed him to develop discretization methods for systematically exploring the possible control paths, giving an approximate solution to the problem [24]. His method requires $2^{O(k)}$ paths to be explored (k is the number of segments in any coordinate direction of the discretization), which implies that exponential work is required.

The work of Canny, Donald, Reif, and Xavier gives the first provably good polynomial time approximation algorithm for the problem of moving a single point though algebraic obstacles in 3-D, with L_∞ acceleration bounds [4]. That algorithm required the trajectory of the point robot to be within an obstacle-free tube whose radius depended on the velocity. Their algorithm finds a movement with elapsed time of at most $(1+\epsilon)T$, where T is the actual minimum trajectory time. The algorithm makes key use of a nonuniform discretization that depends on the velocity. Jacobs, Heinzinger, Canny, and Paden generalized this result to kinodynamics problems with open-chain manipulators and thus coupled dynamics [14]. The single point kinodynamics is generalized to the L_2 norm using more advanced angular discretization methods to better approximate L_2 bounded movements by Donald and Xavier [11] and independently by Reif and Tate [22]. Furthermore, Donald and Xavier improved the analysis of the L_∞ results to an additive error bound $T+\epsilon$, and generalized the results for L_2 norms to kinodynamics problems with open-chain manipulators with bounded joint torques [11]. The above results are all polynomial time, but with very large polynomial constants, and their complexity is strongly dependent on the velocity bounds allowed.

A key further problem is determining cases of kinodynamic control planning that might be solvable (or at least approximated) essentially in closed form. An example of progress on such restricted problems are the translation force control of a point mass in a 2-D room with polygonal obstacles and with the constraint that the L_∞ norm of the force is bounded by some given value. An exact solution for this problem (called the *race car* problem) is given by Canny, Rege, and Reif [5], by partitioning a feasible path into a bounded number of segments where we can use "bang-bang" control. Their algorithm requires more than polynomial sequential time, but only polynomial space. In general, for L_2 norms in dimension $d>1$, the minimal path does not appear to be algebraic.

3 Lower Bounds

In this section we consider polyhedral pursuit games in three dimensions, where the velocity of each player is bounded (as measured by the L_2 norm of the velocity vector). We will show that even this simple pursuit game is hard for the complexity class EXPTIME, where EXPTIME = $\cup_{c\geq 1}$DTIME(2^{n^c}). We first prove the result for a system where only translations are allowed (no rotations), and then we show how a similar construction for arbitrary movement can be constructed, yielding the lower bound for the more general problem.

To prove lower bounds for the pursuit problem, we construct a problem that simulates a given polynomial space bounded alternating Turing machine (ATM) M. The reader unfamiliar with alternating Turing machines can view them as a simple extension of nondeterministic Turing machines (NTMs). As with nondeterministic Turing machines, an ATM has a set of states, a distinguished start state, a set of tape symbols, and a multiply-defined transition function. Unlike an NTM, we label the states as either existential, universal, accepting, or rejecting. We label *configurations* as accepting or rejecting as follows (the labeling is inductively applied, starting at the configurations in either an accepting or rejecting state):

- Any configuration in an accepting state is an accepting configuration.
- Any configuration in a rejecting state is a rejecting configuration.
- Any configuration in an existential state is accepting if and only if *there exists* a transition to an accepting configuration.
- Any configuration in a universal state is accepting if and only if *all* transitions out of this configuration lead to accepting configurations.

An input string is accepted by the ATM if the starting configuration on that input is an accepting configuration. Notice that if the initial configuration is accepting, then there exists a *strategy* for picking transitions in existential states such that the computation always arrives at the accepting state, regardless of the transitions chosen in universal states. In this way, ATMs are related to game strategies between two players. It should be clear that an NTM is simply an ATM with no universal states.

For any function $S(n)$, let ASPACE($S(n)$) denote the class of languages accepted by alternating Turing machines that use at most $S(n)$ space. From the classic work on alternating Turing machines by Chandra, Kozen, and Stockmeyer, we have the following lemma [7, Corollary 3.5].

Lemma 3.1 If $S(n) \geq \log n$, then ASPACE($S(n)$) $= \cup_{c>0}$DTIME($c^{S(n)}$).

It follows from this lemma that APSPACE $= \cup_{c>0}$ASPACE(n^c) = EXPTIME, and furthermore that ASPACE($\log n$) $= P$. The former equality is important to our lower bound proof, and the latter is used to show that our approximation algorithms run in polynomial time.

In the following lower bound proof, we simulate an arbitrary polynomial space-bounded alternating Turing machine M. To simplify the explanation, we assume that the ATM M uses only n tape cells; the generalization to an arbitrary polynomial should be obvious.

One possible point of confusion in the proof that follows is that the existential player of the pursuit game simulates *all* moves of the ATM (both existential and universal); the universal player of our continuous game makes the transition *choices* in universal states, and forces the existential player to actually perform the appropriate transition. To avoid problems with terminology, in this section we will refer to the existential player (of the pursuit game) as the *evader*, and the universal player as the *pursuer*.

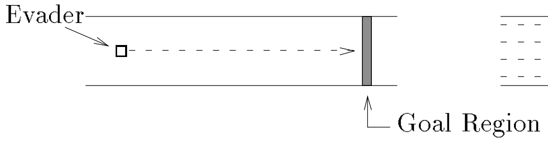
3.1 Basic Geometry and the Encoding of a Configuration

The evader in the construction will be a three-dimensional cube with each side having length 2^{-4n+1} (note that while many dimensions in our construction are exponentially small, only polynomially many bits are required to specify the boundary coordinates). Rectangular tunnels that are exactly 2^{-4n+1} units tall[2] and at most 4 units wide will be the areas in which the evader will travel. The position of the evader along the width of this passage will encode the contents of the ATM's tape. The general idea is to use the distance from a consistently chosen wall of the passage (called the zero wall) to represent the tape contents — if this distance is the binary number $0.a_1a_2...a_n$, then the tape contents are $a_1, a_2, ..., a_n$. Due to the size of the evader, there is clearly enough room to do this (the distance between valid configuration encodings would be 2^{-n}), but a slight modification must be made to include the position of the tape head. This will be discussed in further detail in section 3.4.

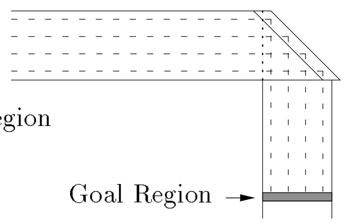
A similar encoding scheme was used by Canny and Reif [6] to represent the values of boolean variables in an instance of 3SAT. Their main result was an NP-hardness proof for 3-dimensional minimum path calculation, and we use several ideas from this earlier work. The most important concept in our lower bound proof is the idea of shortest path classes, modified slightly from the work of Canny and Reif [6]. Consider a polyhedral object (our robot) among a set of obstacles, and a given goal region (possibly disconnected). We wish to move our robot until it touches some point in the goal region. Obviously, if it is possible to reach the goal region, there is some minimum time T required to do so; the set of all paths to the goal region that take time T (there can easily be more than one) is the shortest path class for this problem instance.

In all of our problems, the robot is a small cube (the evader), and the goal region will be a set

[2] In the descriptions of this section, "tall" refers to an object's length in the z-coordinate direction, "wide" refers to the object's length in the x-coordinate direction, and "deep" refers to the object's length in the y-coordinate direction.



(a) Simplest Case



(b) Folded Turn

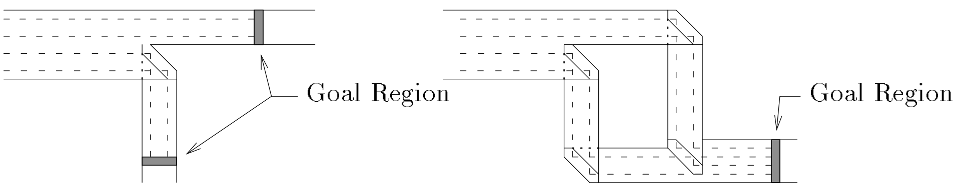
Figure 1: Some Basic Constructions

of cross-sections of the evader's passages. For example, consider the simplest case of a straight, unobstructed passage with the goal region being the end of the passage (see figure 1a for a top view). Obviously, the only path in the shortest path class is the straight-line path shown in the figure. When the evader follows this path, the distance to the walls of the passage is preserved.

We will be defining sets of obstacles (called traps) such that any winning strategy for the evader involves following a shortest path through the trap. The position of the evader in the cross-section of its passageway reflects the configuration of the ATM, and we have seen how this is preserved in shortest-path classes along a straight section of the evader's passage. What if we want the evader to change the direction of its motion? For this we use the "folded turn" shown in figure 1b. Imagine taking a piece of flat ribbon (representing the evader's passage) and folding it at a 45° angle — the top view should be like that shown in figure 1b. Several shortest path classes are shown in the figure, representing various starting positions for the evader. The construction we use cannot be exactly as visualized by folding a flat ribbon, since the passages (and walls) must have some finite thickness. The actual construction consists of a 45° slot cut in the floor of the original passage, and a vertical drop to a second-level passage (below the first). This second passage is oriented at a right angle to the first.

We can also separate the path classes depending on the half of the passage in which the evader is traveling. By using a "folded turn" construction, but having the 45° slot extending only across the bottom half of the passage, and having the goal regions set up across the two resulting passages at an equal distance from the entrance, the path classes will be separated (see figure 2a). The resulting construction is called a "path splitter".

The shuffler is the most important construction, and can be viewed as a combination of the above constructions (see figure 2b). The path splitter divides the top and bottom path classes, and a folded turn on the top half of the passages leaves both halves on the same level. Both passages then take a folded turn to travel horizontally, with the top half of the passages dropping down to



(a) Path Splitter (b) Shuffler

Figure 2: More Basic Constructions

the bottom half with a slight offset from the bottom path classes. Given a discrete set of valid starting positions for the evader, this effectively interleaves (or shuffles) the path classes, as shown in the figure. There is a small technical problem with this construction; namely, we would like the paths for all starting positions to have the same length. Unfortunately, while all paths starting in the lower half of the passage have the same length, this length is slightly shorter than the paths starting in the upper half of the passage. To alleviate this problem, the lower-half path classes are lengthened by adding an additional vertical "jog" (a vertical drop followed by a vertical rise) after the path splitter. With this addition, it should be clear that the shuffler works as desired.

The effect of the shuffler is to change the distance of an evader from $0.a_1a_2...a_n$ (as measured from the zero wall and written in binary fixed point) to $0.0a_2a_3...a_na_1$. This function is vital for testing individual bits of the configuration. As in the paper of Canny and Reif [6], the shuffler halves the width of possible positions for our evader; this was a major problem that limited the time of the simulation in [6] to polynomial. However, in the current situation the presence of the pursuer allows us to overcome this problem (this will be explained further in section 3.4).

The pursuer will be a rectangular box 5 units wide, 2^{-4n} units tall, and 2^{-4n} units deep. Since rotations are not allowed, we see that the pursuer cannot travel in the evader's passage (since the passage is at most 4 units wide), and by making the pursuer's passages 2^{-4n} units deep, the evader will not be able to travel in the pursuer's passages (since the evader is 2^{-4n+1} units deep). In this way, we ensure that there are only a few spots of contention between the pursuer and evader — namely, those places where the pursuer's passage intersects with the evader's passage.

The actual velocity bounds on the pursuer and evader (v_p and v_e, respectively) are not important, but the ratio of the two bounds will determine certain elements of the construction. For the sake of concreteness, we will let $v_p = 10v_e$.

3.2 Basic Form of the Proof

We describe a polyhedral environment that has a polynomial size binary encoding. This environment will be constructible in $O(\log n)$ space by a deterministic TM. We will show that the players will be forced to play essentially in a discrete manner that simulates the given ATM M, or else they will immediately lose the game. We will have a set of obstacles (called the state box) associated with each state of the ATM M. The initial positions for the pursuit game players are at the entrance of the state box associated with the initial state of M, and the position of the evader across the width of its passage encodes the input of the ATM (i.e., the initial tape contents). The goal state of the evader is in the box associated with the accepting state.

The proofs that follow show that for any winning strategy, the only valid paths through each state box correspond to valid transitions of the ATM. By induction, any winning strategy will reach the goal position (corresponding to the accepting state of the ATM) by a series of valid state transitions; therefore, there is a winning strategy if and only if the ATM accepts. The canonical strategy for any accepting ATM reflects the appropriate sequence of existential moves of the ATM.

3.3 Traps

The key component in the lower bound construction is the concept of a *trap*. A trap is a specific region where the evader will become trapped if the shortest path through the region is not taken. After being trapped, there will be no way for the evader to reach the goal position.

The basic trap is illustrated in figure 3. The "evader move box" is some type of basic construction, such as a shuffler. The pursuer's passage starts below the level of the evader's passage, and continues through the evader's passage (the horizontal strip in figure 3); when this passage reaches a certain height, it takes a 90 degree turn to stretch horizontally to a position after the evader's move box. At this point, the passage turns down and continues through the evader's passage. The length of the pursuer's passage is carefully chosen (and set by the height it rises over the evader's passage) so that the pursuer's shortest path from the entrance intersection to the exit intersection takes *exactly* the time of the evader's shortest acceptable path between the intersections. We call the time required by the pursuer to go between the two intersections with the evader's passage the *threshold time* of the trap.

A valid starting position for a trap is as follows. The pursuer is in its passage with its top flush with the floor of the evader's passage. The evader starts at a position following the entrance intersection, with its trailing edge flush with the edge of the intersection. The remaining degree of freedom for the evader's position is arbitrary (in fact, it will encode the ATM's tape contents). Note that in this position the pursuer and evader are actually touching at an edge; however, there

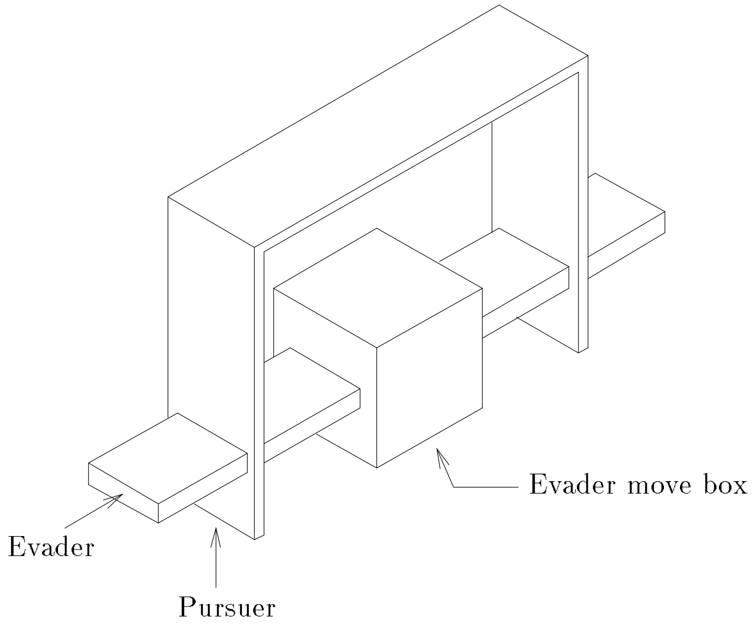


Figure 3: Basic Trap

is no collision as the volume of the intersection is zero.

Theorem 3.1 There exists a strategy for the pursuer such that from a valid starting position, the evader can leave a trap safely if and only if the time it takes to pass the exit intersection is no more than the threshold time of the trap.

Proof: The *state* of the pursuit game at any time can be completely specified by the positions of the players; to denote the state at time t, we write $\mathbf{s}_t$. Let S denote the set of all valid states. For a specific trap, we define the function $\Phi_e : S \to \Re$ that maps states to an amount of time. Specifically, for any state $\mathbf{s}$ (where the evader is in the trap), let t_m be the minimum time required by the evader to travel to the entrance intersection and first become flush with it. The function Φ_e is defined by $\Phi_e(\mathbf{s}) = t_m$ for all such position-time pairs — this may be a fairly difficult function to compute, depending on the complexity of the evader move box, but we are not concerned with the complexity of the pursuer's strategy. Notice that if $\mathbf{s}_0$ is a state with the evader in a starting position for the trap (as described above), then $\Phi_e(\mathbf{s}_0) = 0$. The function $\Phi_p : S \to \Re$ is defined similarly, but for the pursuer; specifically, $\Phi_p(\mathbf{s})$ is the minimum time required for the pursuer to reach its starting position in the trap.

Now we describe a strategy for the pursuer such that the evader will become caught in the trap if its path through the trap takes longer than the threshold time. Notice that if both players travel as fast as possible through the trap, then at all times, $\Phi_e(\mathbf{s}_t) = \Phi_p(\mathbf{s}_t) = t$. The idea behind the pursuer's strategy is as follows: if the evader strays from a minimum distance path, some additional time is available for the pursuer to cover more ground than the evader; in this way, the pursuer can reach the exit intersection before the evader leaves the trap. Obviously, we don't want the pursuer to get *too far* ahead of the evader — otherwise, the evader could exit the trap by reversing its course and leaving through the entrance. Specifically, the pursuer's strategy is to travel forward in its passage as fast as possible, as long as $\Phi_p(\mathbf{s}_t) \leq \Phi_e(\mathbf{s}_t) + \frac{2^{-4n}}{v_e}$. If such a time is ever reached where $\Phi_p(\mathbf{s}_t) = \Phi_e(\mathbf{s}_t) + \frac{2^{-4n}}{v_e}$, then the pursuer simply imitates the progress of the evader, and this equality is maintained.

We now show that the above strategy for the pursuer has the property that the evader can leave the trap only by taking a minimum time path to the exit. First, notice that the evader cannot leave the trap past the entrance — if the evader ever starts traveling backwards, then the pursuer will maintain the equality $\Phi_p(\mathbf{s}_t) = \Phi_e(\mathbf{s}_t) + \frac{2^{-4n}}{v_e}$. When the evader first becomes flush with the entrance intersection we have $\Phi_p(\mathbf{s}_t) = \frac{2^{-4n}}{v_e}$, so the pursuer is able to collide with the evader before the evader can leave the trap. This is because from a position with $\Phi_e(\mathbf{s}_t) = 0$, since the evader has length $2^{-4n+1} = 2 \cdot 2^{-4n}$ and the entrance intersection has length 2^{-4n}, it takes the evader at least $3\frac{2^{-4n}}{v_e}$ time to travel completely past the entrance intersection and out of the trap, but it takes the pursuer at most $\frac{1}{3}$ of this time to cross the evader's passage (colliding with the evader). A similar argument shows that if the evader ever strays from a minimum distance path (so $\Phi_p(\mathbf{s}_t) > \Phi_e(\mathbf{s}_t)$), then the evader cannot leave the exit of the trap without colliding with the pursuer. ∎

Thus if the threshold time is equal to the shortest path time of the evader, then the evader *must* follow a shortest path. However, in the final construction, many traps will be connected together, so how can we be sure that the pursuer cannot improve its strategy by following a path backwards in its passage? In other words, with the above pursuit strategy, we can guarantee that the evader moves forward as quickly as possible — how can we guarantee that the pursuer will do the same? This is easily done by making the pursuer go forward in order to block a path to the goal for the evader. This idea is generalized in the notion of a "forced decision trap".

A top view of the forced decision trap is shown in figure 4 — only the evader's passage is shown in the figure, and the dotted boxes correspond to intersections with the pursuer's passage. The actual 3-dimensional construction is quite complex, and a rough drawing is shown in figure 5. This trap has a single entrance and two exit passages for the evader. Inside the trap (but not shown in figure 4), the pursuer's passage also splits into two separate paths, and the evader must pick which exit to take according to the pursuer's choice. The evader must choose the correct exit *and* take

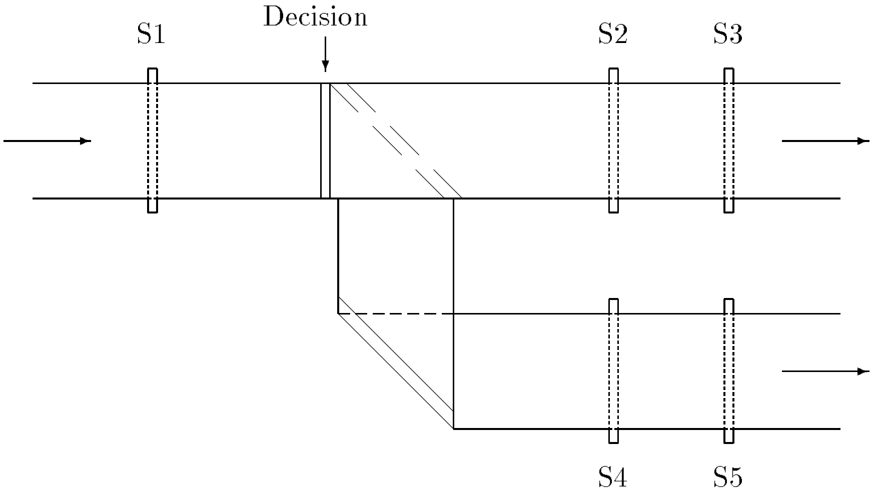


Figure 4: Forced choice trap

the shortest path, or else it will be caught in the trap.

The valid starting position is the same as the basic trap, with respect to slot S1. The evader's passage is a straight passage with a slot cut in its floor (labeled "decision" in figure 4). The passage under the decision slot makes two "folded turns", and continues as the bottom exit of figure 4.

Shortly after the pursuer's passage passes through slot S1, it forks into two passages. One path passes through S2 and then S5, and the other passage goes through S4 and S3. The requirement on the first passage is that the shortest time from slot S1 to S5 is exactly the time it takes the evader to travel from slot S1 to S5. Furthermore, the time required to travel from S1 to S2 must be slightly greater (see the proof of theorem 3.2 for the exact requirements) than the time for the evader to travel from S1 to the decision slot. The second passage has analogous requirements.

When the pursuer takes the passage to S5, the correct path for the evader is to travel to the lower exit along its shortest path. Similarly, the evader must take its shortest path to the upper exit if the pursuer takes the passage to S3.

Theorem 3.2 The evader can leave the trap if and only if it follows the canonical shortest path to the appropriate exit (as chosen by the pursuer).

Proof: In this proof we will assume that the pursuer wants to force the evader to take the bottom exit of figure 4. In such a case, the pursuer stays entirely in its passage between slots S1, S2, and S5. The case where the pursuer forces the evader to take the top exit is almost identical to the case presented below, so is not explicitly given here.

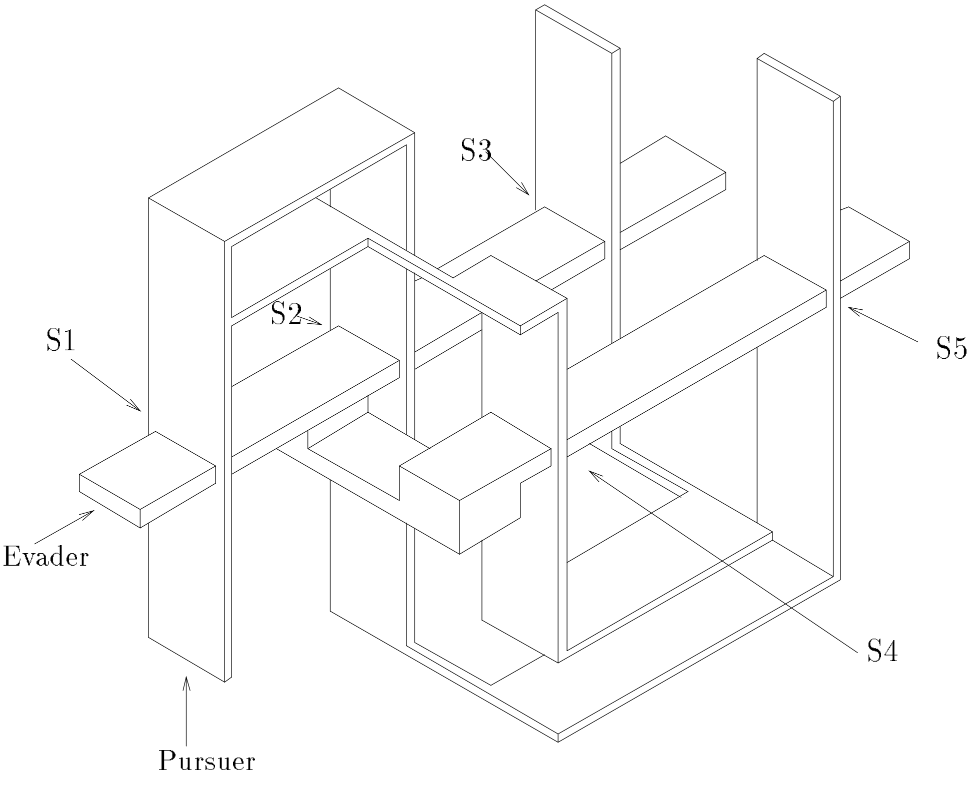


Figure 5: 3-D Rendering of Forced Decision Trap

As in the proof of theorem 3.1, let $\Phi_p(\mathbf{s})$ denote the minimum time required for the pursuer to travel from its position in state $\mathbf{s}$ to its start position in the trap. It is important to remember that regardless of the moves made by the evader, the pursuer stays *only* in the passage from S1 to S2 and S5. We define $\Phi_e(\mathbf{s})$ a little differently than in theorem 3.1. In particular, let t_d be the time required for the evader to reach a position directly above the decision slot. Since the correct path for the evader is to the bottom exit, label all states with the evader to the right of the position over the decision slot and to the left of slot S2 as "bad states". Now define $\Phi_e(\mathbf{s})$ to be the same as in theorem 3.1 whenever $\mathbf{s}$ is not a bad state; in other words $\Phi_e(\mathbf{s})$ is the minimum time required for the evader to travel back to S1. On the other hand, if $\mathbf{s}$ is a bad state then set $\Phi_e(\mathbf{s}) = t_d$.

Now we describe the timing requirement for the pursuer's passage to slot S2: the minimum time required for the pursuer to travel from its start position to a position entirely blocking the evader's passage at S2 must be exactly $t_d + \frac{2^{-4n}}{v_e}$. This requirement is easily set by adjusting how high the pursuer's passage rises above slot S1.

The pursuer's strategy is exactly as in theorem 3.1: the pursuer travels forward in its passage as fast as possible while maintaining $\Phi_p(\mathbf{s_t}) \leq \Phi_e(\mathbf{s_t}) + \frac{2^{-4n}}{v_e}$. If the evader never enters a bad state, then the set of actions is exactly like a basic trap, so the evader must take its shortest path by theorem 3.1. If the evader ever enters a bad state, then the function $\Phi_e(\mathbf{s})$ remains constant for some amount of time. During this time, the pursuer can still travel, insuring that $\Phi_e(\mathbf{s}) > \Phi_p(\mathbf{s})$; as in theorem 3.1, once this inequality is achieved it can be maintained, and the evader cannot leave by either the entrance or the bottom exit. Clearly, when the evader reaches slot S2, then we have had time to reach the point where $\Phi_p(\mathbf{s}) = \Phi_e(\mathbf{s}) + \frac{2^{-4n}}{v_e}$; in other words, the pursuer will fully block the top exit before the evader reaches S2, so the evader is completely caught in the trap.

With the above strategy, the *only* way for the evader to leave the trap is for it to take its shortest path to the bottom exit. Clearly, the case for forcing the evader to take the top exit is almost identical, so is not presented here. ∎

As a special case of the forced decision trap, we can connect the top exit directly with the goal position through a tunnel that the pursuer cannot enter. This ensures that the pursuer keeps moving forward on its path, since it must force the evader to take the bottom exit. By placing these special constructions at positions following each basic trap, we guarantee that if both players follow their best strategy, then they both move forward through the construction as quickly as possible. The forced decision trap will also be used to simulate universal states of the ATM.

3.4 ATM Transitions

Consider the following method of representing the tape contents: the 0's and 1's on the tape correspond to a number in base four notation (not all base four numbers will correspond to valid

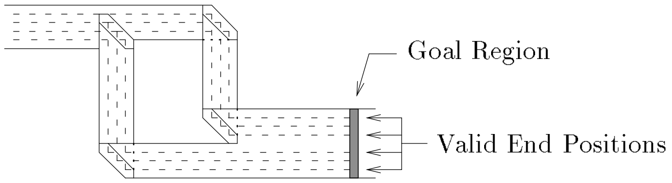


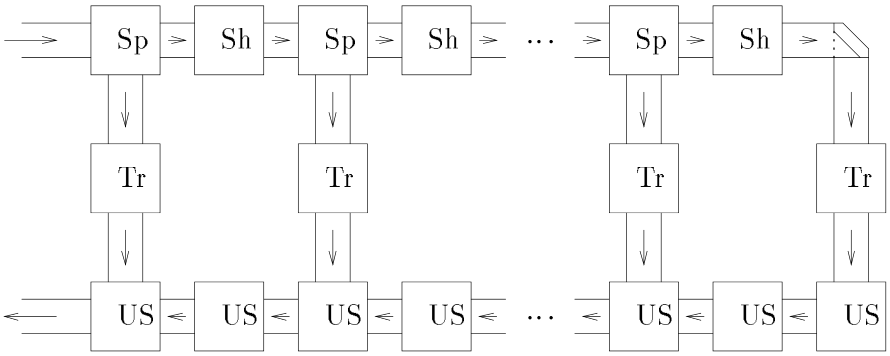
Figure 6: Bad Cases for Unshuffler

tape configurations). In the position that is currently being scanned, the digit is changed from 0 or 1 to 2 or 3, respectively. The distance from a consistently chosen wall (the zero wall) to the evader encodes this representation. Let d denote this distance. Then the base four tape configuration representation r is related to d by $r = d2^{2n-2}$.

We can use the shuffler (figure 2b) to examine bits of the configuration, but only bits in even numbered positions will actually represent symbols on the tape; in the odd positions, a 1 marks the position of the tape head, while all other odd-position bits are 0. The path splitter can be used to "peel off" the configuration when it is shifted to the currently scanned tape position. Traveling through the shuffler backwards has the undesirable side-effect of doubling the number of shortest path classes (but also doubles the width of possible positions for the evader, which is good). In other words, for most input positions of the evader in the reversed shuffler there are two different shortest paths to the exit slot, only one of which is the correct unshuffling path (see figure 6). The incorrect path will always end at a position that is not a valid encoding of a base 4 number. This problem is impossible to overcome in the shortest path proof of [6]. Fortunately, the addition of a pursuer allows us to fix this problem by using a forced decision trap — one side will go to a verifier for the tape configuration, so any winning strategy must do the unshuffle correctly, or else the pursuer could force the evader into the verifier where it will become trapped for not encoding a valid tape configuration.

Lemma 3.2 Assuming the distance d to the zero wall is an integer multiple of 2^{-4n+2}, there exists a trap such that only valid encodings of base four numbers can escape the trap (i.e., only when the evader distance is a multiple of 2^{-2n+2}).

Proof: The basic block is a modified version of the shuffler; it differs from the shuffler in that the final stage (combining the top and bottom halves of the path splitter) does *not* have the offset required by the shuffler. If the evader enters this box at distance $d = 2k + d'$ from the zero wall for $k \in \{0, 1\}$ and $0 \leq d' < 2$, then it leaves the box at distance d' from the zero wall. In other words,



Sp: Path Splitter
Sh: Shuffler
Tr: Transition Block
US: Unshuffler

Figure 7: Block diagram for an entire tape transition function.

the valid positions from the upper half of the evader's passage are overlapped with valid positions in the lower half.

Repeating this construction $2n$ times (for geometrically decreasing passage width), the evader will leave the final exit at distance zero if and only if it entered the construction in a valid position. Since the evader enters at a multiple of 2^{-4n+2}, all *invalid* positions will be at a distance greater than 2^{-4n+2}. By placing a wall in front of all positions further than 2^{-4n+2} from the zero wall and enclosing this structure in a basic trap, all invalid positions will take too much time (since they have to go around the wall) and get caught in the trap. ∎

To perform a state transition, the tape representation is shuffled until the current tape cell is found (i.e., the evader passage is shuffled until the evader is in the upper half of the passage). To accomplish this, consider alternating path splitters with shufflers (as in the top row of Figure 7). The splitters strip off the evader when the tape head position is found, and then the currently scanned tape symbol is the most significant bit of the evader's distance to the zero wall, so it can be easily tested, set, or reset. Even after the maximum number of shuffles ($2n$), there is still a 2^{-4n+2} spacing between valid positions for the evader, so it is still a simple matter to distinguish between different configurations. Since, after being split off of the main passage, a tape update simply involves adding a constant value to the configuration encoding, a transition is easily performed by simply shifting the evader's position in its passage[3] (actually, the passage is shifted, and the

[3]This changes only the tape contents and tape head position. The remaining part of a transition — the state

evader maintains a straight-line path), followed by the correct number of unshufflers. This entire construction is shown in Figure 7. Following all the unshufflers is a forced decision trap with one exit linked to a verifier as described in the lemma above (the output of the verifier is a passage to the goal that the pursuer cannot enter, so that it would be a bad strategy for the pursuer to force a correctly unshuffled evader into the verifier) — this ensures that all unshufflers work correctly. The total number of gadgets required is n shufflers, n path splitters, $2n$ unshufflers, a forced decision trap, and a verifier. The total number of bits required to encode these constructions is clearly polynomial in n.

A set of gadgets is built for every state in the ATM. If the state is existential and there are k possible transitions out of this state, then the incoming evader passage forks into k passages. Forced decision traps are placed at the end of each of the k passages, with the bottom exit of each trap linked to the goal position. In this way, we give the evader a free choice of which passage (i.e., transition) to take, and the forced decision trap makes the pursuer chase the evader into the chosen transition. The top exit of each forced decision trap is connected to a gadget that performs a transition as described above, and the output of each transition goes to the corresponding next state.

If the state is universal with k next states, then construct a $\lceil \log k \rceil$ depth tree of forced decision traps with all but k leaves (those corresponding to valid transitions) linked directly to the goal. Each non-goal exit is followed by a construction that performs a transition out of the universal state, and the particular transition performed can be chosen by the pursuer making decisions at all the internal nodes of the tree (recall that these are forced decision traps). Since all "extra" leaves were connected directly to the goal, the pursuer will never force the evader to travel to one of these invalid leaves.

Notice that in existential states, the evader has a free choice of which path to take, but for a winning strategy it must make a choice compatible with *all* possible future universal options (since the pursuer can arbitrarily force any choice in the universal states). We have proved the following theorem.

Theorem 3.3 For any polynomial space bounded ATM and n-bit input x, a pursuit game with no rotations can be constructed such that a winning strategy exists if and only if the ATM accepts x. The pursuit game has a polynomial length binary encoding, and can be computed by a Turing machine in $O(\log n)$ space.

To extend the proof to a lower bound when rotations are allowed, consider a pursuer as above, but with a groove cut in the top. The pursuer passages then have tracks that fit into the pur-

change — is accomplished by connecting the exit of this entire construction to the input of the next state.

suer's groove and make rotations impossible. Re-examination of the above construction shows that allowing rotation of the evader does not affect the lower bound proof.

Corollary 3.1 For any polynomial space bounded ATM and n-bit input x, a general pursuit game can be constructed such that a winning strategy exists if and only if the ATM accepts x. The pursuit game has a polynomial length binary encoding, and can be computed by a Turing machine in $O(\log n)$ space.

The following corollary follows from the fact that APSPACE=EXPTIME.

Corollary 3.2 Any algorithm that solves the decision problem for the polyhedral pursuit game (either with or without rotations) must take at least exponential time in the worst case.

4 Approximation Algorithms

In this section, we look at polyhedral pursuit games with various types of dynamics bounds, and develop approximation algorithms for these games. The closeness of the approximation (as defined below) is given by a parameter $\epsilon > 0$. We are also given a deadline time for the pursuit game that is bounded by a polynomial in $1/\epsilon$.

Given any rational number ϵ, we call a strategy ϵ-safe if the strategy will always keep distance ϵ between the evader and both the pursuer and all obstacles. In this section we give an algorithm which, when given a pursuit game and a safety margin ϵ, will always find a winning strategy if there exists an ϵ-safe strategy. If a winning strategy exists, but there is no ϵ-safe strategy, then the algorithm may or may not find a winning strategy (but will never give a bad strategy).

Such approximation algorithms exist for problems where either the velocity or both velocity and acceleration are bounded, and the bound can be on either the L_2 or L_∞ norm (although a slight concession must be made on the dynamics bounds in the L_2 case). For each variant of the problem, we have a different closeness lemma (this is lemma 4.1 below, for bounded L_∞ norm of velocity), but the relation to the continuous pursuit games is the same in every case. In the first section below, we present proofs for the simplest case: bounded L_∞ norm of velocity, with no bound on acceleration. The other cases are similar and involve "closeness" proofs (which we give at the end of this section) that are very similar to the tracking lemmas in approximately optimal kinodynamic planning (see [22] and [11]).

4.1 Bounded L_∞-norm velocity

The following discrete game will be used on a discretization of the geometry of the continuous game. Since the reachability sets for the players may be different (due to the different shapes of the players), we need a way of marking which players can follow which edges. This is the purpose of the labeling function below.

Consider the following discrete game. The input is a graph where each edge e has a label $l_e \subseteq \{0,1\}$. Two players (player 0 and player 1) start at given vertices, and player p is allowed to traverse edge e if $p \in l_e$. The game consists of rounds where each player traverses a valid edge for that player, and we wish to know if there is a strategy for player 0 so that it can reach a given goal vertex while never coming "close" (within two steps on the graph) to player 1. It should be clear that this game can be decided in ASPACE($\log n$) for an n vertex graph.

We show that slight variants of this game can be constructed to solve the ϵ-safe approximation version of pursuit games. In d-dimensions, the graph will have $O(\left(\frac{1}{\epsilon}\right)^d)$ vertices for bounded L_∞ norm velocity, and slightly more (but still $\left(\frac{1}{\epsilon}\right)^{O(d)}$) for the more complex pursuit games. We now present the case for pursuit games with a bound on the L_∞-norm of the players' velocity.

If we are given the starting configuration for a pursuit game, and bounds on the L_∞ norm of the velocity for the evader and the pursuer (denote the bounds by v_e and v_p, respectively), we superimpose a regular grid G with grid-spacing g on the d-dimensional environment. We label each grid-point by a d-tuple of integers $(x_1, x_2, ..., x_d)$. A graph is constructed on the grid by connecting every point $(x_1, x_2, ..., x_d)$ to points $(y_1, y_2, ..., y_d)$ with $|x_i - y_i| \leq 1$ for all $i = 1, 2, ..., d$. (This is just a d-dimensional grid-graph with diagonal edges added.) An arbitrary point on each player is chosen; a player is at grid-point p when the chosen point of the player is at the grid-point p. The edges of the graph are labeled according to whether the path joining the two endpoints is free of obstacles for each player.

We will choose a sufficiently small grid-size so that the discrete game is a good approximation of the continuous game. First, we show that for sufficiently small grid-size, there is always a good strategy for the evader (traveling on the grid) against a *continuous* pursuer.

Lemma 4.1 Assume $g \leq \frac{2}{9}\epsilon$. Then if there is an ϵ-safe strategy for the continuous evader, there is a $3g$-safe strategy that travels only between grid-points.

Proof: Consider any path for the pursuer, and the corresponding ϵ-safe path (given by the ϵ-safe strategy) for the evader. We will approximate the continuous evader's path with a path traveling between grid-points. It takes the evader exactly $\tau = \frac{g}{v_e}$ time to traverse any edge of the grid-graph. By induction, it is easy to show that there is a grid-path that is no further than $v_e\tau = g$ away from the continuous path at all times $k\tau$ for k any integer. Furthermore, since the path is a close approximation at these discrete times, it is easy to show that the grid-path is no further than $v_e\tau + v_e\frac{\tau}{2} = \frac{3}{2}g$ at *all* times.

In particular, since the pursuer is at least ϵ away from the evader at all times, the distance from the pursuer to the grid-path evader must be at least $\epsilon - \frac{3}{2}g \geq \frac{9}{2}g - \frac{3}{2}g = 3g$. ▮

When the pursuer is restricted to traveling on the grid, there is a problem with the chosen grid-size being incompatible with the pursuer's velocity bound. For example, if the bounds are such that $v_p = \frac{3}{2}v_e$, then for each edge traversal of the evader, the pursuer can traverse one and a half edges. This does not fit into the simple discrete game defined earlier, so we make the following change. An additional game parameter s (a rational number called the scaling factor) is introduced, and the effect of this parameter is that the pursuer will make s moves for each move of the evader. For non-integer values of s, the meaning of this is unclear — if we are in round r of the game, we actually let the pursuer make $\lfloor rs \rfloor - \lfloor (r-1)s \rfloor$ moves. Using this scheme, the pursuer will always be within one grid-point of the place it would be if fractional moves along edges were allowed.

In our simulation, we let $s = \frac{v_p}{v_e}$. The following lemma ensures us that restricting the pursuer to a grid-path is not a great advantage for the evader.

Lemma 4.2 If both players are restricted to making movements between grid-points, then any winning strategy for the evader will also give a winning strategy against a continuous pursuer.

Proof: Consider any continuous path for the pursuer. If fractional edge traversals were allowed, then we could make an approximating path for the pursuer just as we did for the evader in lemma 4.1. This path is always within $\frac{3}{2}g$ of the continuous path, but due to the discretization of fractional moves, an additional error of g may be introduced. Thus at all times, the grid-path pursuer is within $\frac{5}{2}g$ of the continuous pursuer.

By the definition of the discrete game, we know that the distance between the discrete versions of the players is at least $3g$. Thus the continuous pursuer must be at least $\frac{1}{2}g$ away from the evader (i.e., the evader is not captured by the continuous pursuer). The strategy for the evader against a continuous adversary is therefore to make exactly the same moves as the discrete player would make against the discrete approximation of the continuous pursuer. This is clearly a winning strategy. ▮

The combination of the two preceding lemmas gives the proof of correctness for our approximation algorithm.

Theorem 4.1 If $g \leq \frac{2}{9}\epsilon$, then the discrete game will always find a winning strategy when there is an ϵ-safe strategy for the original game. Furthermore, any winning strategy found in the discrete game is also a winning strategy for the continuous game. The sequential time complexity of the approximation algorithm is $(n/\epsilon)^{O(1)}$.

Proof: By lemma 4.1, if there is an ϵ-safe strategy, then there is a $3g$-safe strategy that only uses grid moves. Restricting the pursuer to the grid means that at all times the pursuer is at least 3 edge traversals away from the evader — this is exactly the condition we need to satisfy for a winning strategy in the discrete game. The second claim in the theorem is exactly lemma 4.2.

The complexity of this algorithm is exactly that of the discrete game, with one minor modification. We only calculate grid-point adjacencies when a player is at the grid-point in question. To determine the adjacencies, we only need to do simple calculations on the obstacle descriptions — this can be done in $O(\log n)$ space, so the resulting complexity is ASPACE$(d \log(1/\epsilon) + \log n)$, or $(n/\epsilon)^{O(d)}$ sequential time. For constant d, this is simply $(n/\epsilon)^{O(1)}$. ∎

Consider a generalization of this problem where the obstacles are allowed to move with constant velocity. The location of all obstacle coordinates can then be computed by a simple linear function of time, and it takes no more space than the original algorithm to compute vertex adjacencies.

Corollary 4.1 Given a pursuit game where obstacles are allowed to move with constant velocity, if $g \leq \frac{2}{9}\epsilon$, then we can approximately compute a winning strategy (in the sense of the last theorem) in sequential time $(n/\epsilon)^{O(1)}$.

In the following sections, we describe the discretization required for other forms of dynamics bounds, and prove closeness lemma in each case.

4.2 Bounded L_∞-norm velocity and acceleration

Now we consider a pursuit game where the L_∞-norm of both velocity and acceleration are bounded. We use v_e and a_e to denote the velocity and acceleration bounds for the evader; v_p and a_p represent the velocity and acceleration bounds for the pursuer. The "grid" produced is not a regular grid in position space, as it was in the previous pursuit game. Instead, we construct a regular grid in velocity space, and the position grid consists of points corresponding to moves on the velocity grid. This is exactly the method used in kinodynamic planning, and we use the results of that work here; for more details on the exact method, see [4] and [11].

The grid construction (from [4]) is specified by the discrete time-step τ (along with the value of a_e). The following closeness lemma for this game is stated in terms of this time-step.

Lemma 4.3 Assume $\tau \leq \min(\frac{\epsilon}{20 v_e}, \frac{v_e}{2a_e})$. Then if there is an ϵ-safe strategy for the continuous evader, there is a $\frac{3}{4}\epsilon$-safe strategy that travels only between grid-points.

Proof: The "Strong Tracking Lemma" of [11] states that if $\tau \leq \min(\frac{\eta_x}{5 v_e}, \frac{v_e}{2a_e})$, then for any continuous trajectory meeting velocity and acceleration bounds v_e and a_e, respectively, there exists a grid trajectory that is always within distance η_x of the continuous trajectory. The lemma above follows by setting $\eta_x = \frac{\epsilon}{4}$. ∎

Now, of course, we must consider what happens when we discretize the continuous pursuer's trajectory. We can guarantee that the grid pursuer (with the scaling factor as before) stays within $\frac{\epsilon}{2}$ of the continuous pursuer's trajectory as long as $\tau \leq \min(\frac{\epsilon}{20v_p}, \frac{v_p}{2a_p})$. Notice that this means that when approximating both continuous players on the grid, there is always at least $\frac{\epsilon}{4}$ distance between the players, so the approximating trajectories correspond to a winning strategy for the evader. This fact, combined with the preceding lemma, gives the following theorem.

Theorem 4.2 Assume $\tau \leq \min(\frac{\epsilon}{20v_e}, \frac{v_e}{2a_e}, \frac{\epsilon}{20v_p}, \frac{v_p}{2a_p})$. Then the discrete game will always find a winning strategy when there is an ϵ-safe strategy for the continuous game. Furthermore, any winning strategy found in the discrete game is also a winning strategy for the continuous game. For a constant number of dimensions d, the sequential time complexity of the approximation algorithm is polynomial in $\frac{n}{\epsilon}$ and the parameters v_e, a_e, v_p, and a_p.

Proof: The size of the grid is given in [11], and it is upper bounded by

$$\left[\left(\frac{2v_{\max}}{a_{\max}\tau} + 1\right)\left(\frac{D}{a_{\max}\tau^2} + 1\right)\right]^d,$$

where $a_{\max} = \max(a_e, a_p)$, $v_{\max} = \max(v_e, v_p)$, and D is the diameter of the robot world. Since the time of our simulation is bounded by a polynomial in $(\frac{1}{\epsilon})^{O(1)}$, we can bound D by $\frac{v_{\max}}{\epsilon^{O(1)}}$. Furthermore, $\frac{1}{\tau}$ is polynomial in $\frac{1}{\epsilon}$, v_e, a_e, v_p, and a_p, so the result is an algorithm polynomial in these values, as stated in the theorem. ∎

4.3 Bounded L_2-norm velocity

As in kinodynamic planning, bounding the L_2-norm for dynamics bounds adds additional problems to approximation algorithms. Specifically, we need to be able to closely approximate the *direction* of motion (or acceleration, in the following section). In kinodynamic planning, we find an approximately optimal trajectory that takes time $(1+\epsilon)T_{\min}$, where $T_{\min}$ is the time required by the optimal trajectory. In the current pursuit game, we cannot allow this extra time because of the interaction with the pursuer — instead, we must allow the evader to exceed its dynamics bounds by a factor of ϵ. In other words, we allow the evader to have velocity as high as $(1+\epsilon)v_e$; the effect of this is identical to allowing the evader to take extra time, without the bad effects of allowing the evader more time.

In this section, we do not require the full power of the kinodynamic tracking lemma, but we use another theorem from [22] to prove the following closeness lemma. We use a regular graph in position space with grid-spacing g, as in the case of bounded L_∞-norm velocity, but only parts of the grid are used. In particular, with a specified discrete time-step τ and error bound ϵ, we can

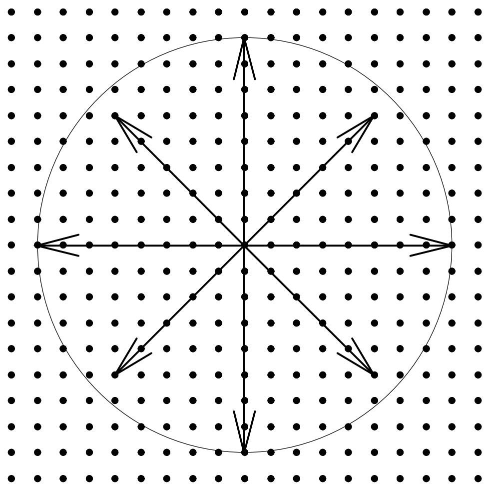


Figure 8: Possible choice vectors in two dimensions for $\epsilon = \frac{1}{2}$.

construct a grid similar to that shown in Figure 8. The arrows (called "choice vectors") denote the possible trajectories over the discrete time step, and the dots represent the underlying square grid (with grid spacing g) — the circle is shown only for reference and has radius $v_{\max}\tau$, so delimits the maximum distance the robot can travel in one time step. The choice vectors are chosen such that the angle between neighboring choice vectors is at most $\arccos(1 - \frac{\epsilon}{4(1+\epsilon)})$.

Assume that we want to track a continuous trajectory $\mathbf{p}_c(t)$. We call a discrete trajectory $\mathbf{p}_d(t)$ a "good approximation" if at all the discrete times $k\tau$ (with $k = 0, 1, 2, ...$), the distance between the two trajectories is bounded by $\|\mathbf{p}_d(k\tau) - \mathbf{p}_c(k\tau)\| \leq v_{\max}\tau$. It has been shown that if $g \leq \frac{\epsilon}{4} v_{\max}\tau$, then for any continuous trajectory meeting the velocity bound $\frac{v_{\max}}{(1+\epsilon)}$ there exists a good approximating trajectory made up of only choice vectors as described above [22, Theorem 3.1]. Using this result, we can prove the following lemma.

Lemma 4.4 Assume $g \leq \frac{\epsilon^2}{3(2+\epsilon)}$. Then if there is an ϵ-safe strategy for the continuous evader, there is a $3g$-safe strategy that travels only between grid-points.

Proof: We use the discrete vectors from [22], as described above, with $v_{\max} = (1+\epsilon)v_e$. By setting $g = \frac{\epsilon}{4} v_{\max}\tau$, and using the fact that the good approximating trajectory is within distance $v_{\max}\tau$ of the exact (continuous) trajectory at all times $k\tau$, it is clear that the good approximating trajectory is within distance $\frac{4g}{\epsilon}$ of the continuous trajectory at all times $k\tau$. As in lemma 4.1, we conclude that there is a discrete trajectory that is within $\frac{4g}{\epsilon} + \frac{1}{2}\frac{4g}{\epsilon} = \frac{6g}{\epsilon}$ of the continuous trajectory at *all* times.

Since the continuous trajectory is ϵ-safe, this approximating trajectory is always at least $\epsilon - \frac{6g}{\epsilon}$ distance away from the pursuer and all obstacles. The proof is completed by noticing that

$$\epsilon - \frac{6g}{\epsilon} \geq \epsilon - \frac{2\epsilon}{2+\epsilon} = \frac{\epsilon^2}{2+\epsilon} \geq 3g. \quad \blacksquare$$

The remainder of the proof of correctness for this case is almost identical to the bounded L_∞-norm velocity case, so is not spelled out here. The result is the following theorem.

Theorem 4.3 If $g \leq \frac{\epsilon^2}{3(2+\epsilon)}$, then the discrete game will always find a winning strategy when there is an ϵ-safe strategy for the original game. Furthermore, any winning strategy found in the discrete game is also a winning strategy for the continuous game. The sequential time complexity of the approximation algorithm is $(n/\epsilon)^{O(1)}$.

4.4 Bounded L_2-norm velocity and acceleration

When the L_2-norm of both velocity and acceleration are bounded, we use all of the ideas from the previous cases, in addition to the full L_2 tracking lemma from [12] (we could also use the tracking lemma of [22], but the bounds of [12] are slightly better). As in the previous L_2-norm case, we must allow the evader to slightly exceed its dynamics bounds; specifically, we allow the approximating evader to have velocity $(1+\epsilon)v_e$ and acceleration $(1+\epsilon)^2 a_e$.

Previously, Donald and Xavier [12, Lemma 6.3] have shown that any continuous trajectory meeting acceleration bound $\frac{a_{\max}}{(1+\epsilon)^2}$ can be approximated by a discrete trajectory with time-step τ and positional error η_x as long as

$$\tau \leq \sqrt{\frac{\eta_x \epsilon}{a_{\max}(48+2\epsilon)}}.$$

Substituting $\eta_x = \frac{\epsilon}{4(1+\epsilon)^2} \leq \frac{\epsilon}{4}$ and $a_{\max} = (1+\epsilon)^2 a_e$, this result implies that there is a discrete trajectory that tracks our evader with positional error bounded by $\frac{\epsilon}{4}$. The result is stated in the following lemma.

Lemma 4.5 Assume $\tau \leq \frac{\epsilon}{2\sqrt{a_e(48+2\epsilon)}}$. Then if there is an ϵ-safe strategy for the continuous evader, there is a $\frac{3}{4}\epsilon$-safe strategy that travels only between grid-points.

The final result for the bounded L_2-norm velocity and acceleration game is stated in the following theorem.

Theorem 4.4 Assume $\tau \leq \frac{\epsilon}{2\sqrt{a_{\max}(48+2\epsilon)}}$, where $a_{\max} = \max(a_e, a_p)$. Then the discrete game will always find a winning strategy when there is an ϵ-safe strategy for the continuous game. Furthermore, any winning strategy found in the discrete game is also a winning strategy for the continuous game. For a constant number of dimensions d, the sequential time complexity of the approximation algorithm is polynomial in $\frac{n}{\epsilon}$ and the parameters v_e, a_e, v_p, and a_p.

5 The Point-Robot Pursuit Game

In many other robotics problems, it can be assumed that the robot is a single point; more difficult problems are reduced to a point-robot problem by growing the obstacles according to the shape of the robot. In the pursuit game, there are two moving robots with possibly different shapes, so this obstacle-growing technique will not work.

In this section, we consider the pursuit game where each player is a single point, and show that this problem is computationally easier than the original pursuit game (assuming that PSPACE is a proper subset of EXPTIME). We bound the velocity (but not acceleration) of each player, and further restrict the pursuer's velocity bound to be at least as high as the evader's velocity bound. Notice that this game is identical to the game used in the lower bound construction of section 3, except that the players are points. A key factor of the lower bound proof is that each player can only travel in its own passage, restricting the points of contention to several discrete locations. We cannot use this construction when the players are points, so the lower bound does not apply to this restriction. In fact, we can show that in this case, the problem is easily reducible to the shortest path problem.[4] We first prove the following theorem, showing the relationship between the point-robot pursuit game and the shortest path problem.

Theorem 5.1 In the point-robot pursuit game described above, there exists a winning strategy for the evader if and only if its fastest path to the goal is quicker than the pursuer's fastest path to the goal.

Proof: Let T_e (resp. T_p) be the minimum time required for the evader (resp., pursuer) to reach the goal. These can easily be calculated from the shortest *distance* path to the goal simply by dividing the distance by the maximum allowed velocity.

First assume that there is no winning strategy for the evader. Then, in particular, there is some trajectory for the pursuer that collides with the evader traveling along its fastest path to the goal. Let the time of collision be t_c. Ignoring the evader, the pursuer could follow its collision path until time t_c, and then follow the remaining segment of the evader's fastest path to the goal. This new path for the pursuer requires time

$$t_c + \frac{v_e}{v_p}(T_e - t_c) \le t_c + T_e - t_c = T_e$$

(recall that the evader's velocity bound is no greater than the pursuer's bound, so $\frac{v_e}{v_p} \le 1$). In other words, if the pursuer can always catch the evader, then the pursuer's fastest path to the goal takes no longer than the evader's fastest path to the goal.

[4]The reduction used is a *Turing reduction*. In other words, we assume that there is an oracle for shortest path, and in particular, our reduction makes only a constant number of calls (two) on the oracle.

Now assume that the purser's fastest path to the goal takes no longer than the evader's fastest path. Then a winning strategy for the pursuer is to simply move to the goal as fast as possible, effectively blocking the evader from the goal. In other words, there can be no winning strategy for the evader.

This proves both directions of the "if and only if" statement in the theorem. ∎

From the above theorem, the following corollary is obvious.

Corollary 5.1 The point-robot pursuit game described above is Turing reducible to the shortest path problem, and only two calls on the shortest path oracle are required.

6 Open Problems

There are many open problems in the area of pursuit games. One of the most interesting questions is to see what type of lower bound can be derived for pursuit games in which the L_∞ norm of velocity is bounded. Notice that in our lower bound construction, the position of the evader along the width of the passage acted as a "memory" of previous moves. With bounded L_∞ norm, the dimension representing the width of the passage is free to move with *no decrease* in the time it takes to travel through a basic trap. This allows the evader to "cheat" by performing invalid state transitions.

Another interesting open problem is to look at exact solutions for restricted games. For instance, many interesting problems have few (or no) obstacles. In addition, the lower bound is for three or more dimensions; are exact solutions possible in two dimensions?

References

[1] J. E. Bobrow. Optimal Control of Robotic Manipulators. Ph.D. Thesis, UCLA, Mechanics and Structures Department, 1982.

[2] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control.* Hemisphere Publishing Corporation, Washington, D. C., 1975.

[3] J. Canny. Some Algebraic and Geometric Computations in PSPACE. In *20th ACM Symposium on Theory of Computing*, pages 460–467, 1988.

[4] J. Canny, B. Donald, J. Reif, and P. Xavier. On The Complexity of Kinodynamic Planning. In *29th IEEE Symposium on Foundations of Computer Science*, pages 306–316, 1988.

[5] J. Canny, A. Rege, and J. Reif. An Exact Algorithm for Kinodynamic Planning in the Plane. In *6th Annual ACM Symposium on Computational Geometry*, pages 271–280, 1990.

[6] J. Canny and J. Reif. New Lower Bound Techniques for Robot Motion Planning Problems. In *28th IEEE Symposium on Foundations of Computer Science*, pages 49–60, 1987.

[7] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1):114–133, January 1981.

[8] K. L. Clarkson. Approximation Algorithms for Shortest Path Motion Planning. In *19th ACM Symposium on Theory of Computing*, pages 56–65, 1987.

[9] G. Collins. Quantifier Elimination for Real Closed Fields by Cylindric Algebraic Decomposition. In *2nd GI Conference on Automata Theory and Formal Languages*, pages 134–183, 1975.

[10] B. Donald. The Complexity of Planar Compliant Motion Planning Under Uncertainty. In *4th Annual ACM Symposium on Computational Geometry*, pages 309–318, 1988.

[11] B. Donald and P. Xavier. A Provably Good Approximation Algorithm for Optimal-Time Trajectory Planning. In *IEEE International Conference on Robotics and Automation*, pages 958–963, 1989.

[12] B. Donald and P. Xavier. Provably Good Approximation Algorithms for Optimal Kinodynamic Planning for Cartesian Robots and Open Chain Manipulators. Technical Report TR-1095, Cornell University, February 1990.

[13] M. Hollerbach. Dynamic Scaling of Manipulator Trajectories. In *Proceedings of the American Control Conference*, pp. 752–756, 1983.

[14] P. Jacobs, G. Heinzinger, J. Canny, and B. Paden. Planning Guaranteed Near Time-Optimal Trajectories for a Manipulator in a Cluttered Workspace. Technical Report ESRC 89-20/RAMP 89-15, Engineering Systems Research Center, University of California, Berkeley, 1989.

[15] T. Lozano-Perez and M. A. Wesley. An Algorithm for Planning Collision-free Paths Among Polyhedral Obstacles. *Communications of the ACM*, 22(10):560–570, October 1979.

[16] C. ÓDúnlaing. Motion Planning with Inertial Constraints. *Algorithmica*, 2(4):431–475, 1987.

[17] C. H. Papadimitriou. An Algorithm for Shortest-Path Motion in Three Dimensions. *Information Processing Letters*, 20(5):259–263, June 1985.

[18] J. H. Reif. Complexity of the Mover's Problem and Generalizations. In *20th IEEE Symposium on Foundations of Computer Science*, pages 421–427, 1979.

[19] J. H. Reif. A Survey on Advances in the Theory of Computational Robotics. In K. S. Narendra, editor, *Adaptive and Learning Systems: Theory and Applications*. Plenum Press, New York, NY, 1986.

[20] J. H. Reif and M. Sharir. Motion Planning in the Presence of Moving Obstacles. In *26th IEEE Symposium on Foundations of Computer Science*, pages 144–154, 1985.

[21] J. H. Reif and J. A. Storer. Minimizing Turns for Discrete Movement in the Interior of a Polygon. *IEEE Journal of Robotics and Automation*, RA-3(3):182–193, June 1987.

[22] J. H. Reif and S. R. Tate. Approximate Kinodynamic Planning Using L_2-norm Dynamics Bounds. Technical Report CS-1990-13, Duke University Department of Computer Science, 1990.

[23] J. Renegar. On the Computational Complexity and Geometry of the First-order Theory of the Reals. Technical Report 853, Cornell University, School of O.R. and I.E., 1989.

[24] G. Sahar and J. M. Hollerbach. Planning of Minimum-Time Trajectories for Robot Arms. In *IEEE International Conference on Robotics and Automation*, pages 751–758, 1985.

[25] H.M. Schaettler. On the Optimality of Bang-Bang Trajectories in R^3. *Bulletin of the American Mathematical Society*, 18:113–116, 1987.

[26] J. T. Schwartz and M. Sharir. On the Piano Movers' Problem: I. The Case of a Rigid Polygonal Body Moving Amidst Polygonal Barriers. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.

[27] M. Sharir and A. Schorr. On shortest paths in polyhedral spaces. In *16th ACM Symposium on Theory of Computing*, pages 144–153, 1984.

[28] K. G. Shin and N. D. McKay. Selection of Near-Minimum Time Geometric Paths for Robotic Manipulators. In *Proceedings of the American Control Conference*, pages 346–355, 1985.

[29] K. Sutner and W. Maas. Motion Planning Among Time-Dependent Obstacles, 1985. preprint.