Chapter 10

Uncertainty

10.1 General Uncertainty Concepts

There are generally two different ways in which uncertainty is introduced into the motion strategy problem:

- **Uncertainty in Prediction:** This form is uncertainty generally arises from our inability to predict a *future* situation. When unpredictability is present, it is generally inappropriate to assume that a precomputed path from an initial configuration to a goal configuration will solve the problem. If unpredictable changes occur, the robot might have to change its course of action during execution. In general, the robot will have to use a motion strategy that includes feedback, allowing its actions to depend on the current configuration or state. Typical problems that involve uncertainty in predictability include: indoor mobile robots, planetary exploration vehicles, robot manipulators, underwater robots, floating blimp robots, target-tracking robots, and an automated car in traffic.

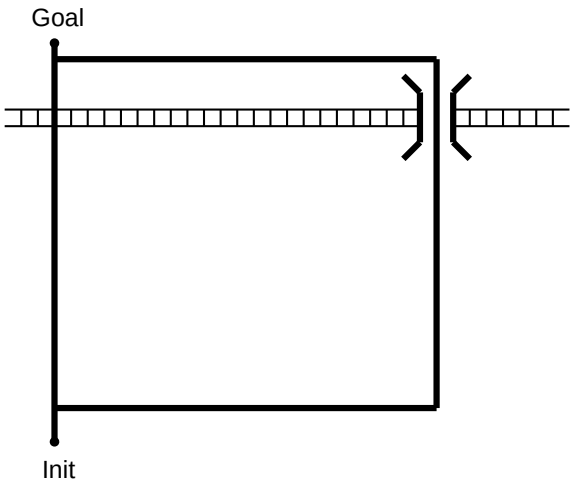
- **Uncertainty in Sensing:** This form of uncertainty occurs if we are unable to determine the *current* situation. When this form is present, a motion strategy often cannot be described directly in terms of configuration or state. The robot must choose actions based on information that is gathered from sensing. Examples include mobile robots, robots with vision systems, and robots in highly-unstructured and partially-known environments.

With either of these forms of uncertainty, there are several possible things that can be uncertain, such as the configuration, the state, the model of the world, or even the model of the robot.

For a given motion motion strategy problem, it might be appropriate to model either one of these uncertainties, or both.

A game against nature A vast set of motion strategy problems that involve uncertainty can be formulated as a game that is played between the robot and "nature." The robot and nature each has a set of actions to choose from. The nature player will generally choose actions that are unknown or unpredictable to the robot. Thus, a robot faced with uncertainties must make decisions that overcome an "opponent's" actions. The robot has a loss or cost function, L, which serves the same purpose as in the forward search method for nonholonomic planning.

To quickly identify the main concepts, consider the toy problem illustrated below. The robot would like to move from the initial position to the goal position, but along the shortest route lies a train crossing. The robot can choose either to go around the train crossing (over a bridge), or it can take the shortest route. There is a chance, however, that the robot would have to wait for a while if the train is coming.



The robot has two actions: go straight or take the longer route. Nature has two actions: it will either prevent a train from coming, or allow a train to come. Since each player has two possible actions, there are four possible outcomes. If the robot goes straight and there is no train, then $L = 1$, but if the train comes, then $L = 4$. If the robot takes the longer route, then $L = 3$, regardless of whether a train comes. This information can be encoded in a matrix,

$$\begin{pmatrix} 1 & 4 \\ 3 & 3 \end{pmatrix},$$

in which the rows correspond to the robot's actions and the columns correspond to nature's actions.

There are two general ways to model nature: *nondeterministic* and *probabilistic*.

Nondeterministic modeling A nondeterministic model means that we have no idea what nature will choose. This usually leads to *worst-case analysis* when designing a motion strategy.

In the example, the robot should take the longer route, because in the worst-case, the train will come, implying $L = 4$. By taking the longer route, $L = 3$, which is always preferred.

Probabilistic modeling A probabilistic model means that nature's actions will be selected according to a probability distribution. This generally allows more information to be exploiting when designing a motion strategy, but in many applications it may be impossible to determine statistically-valid probabilities. The probabilistic model usually leads to *average-case* or *expected-case* analysis.

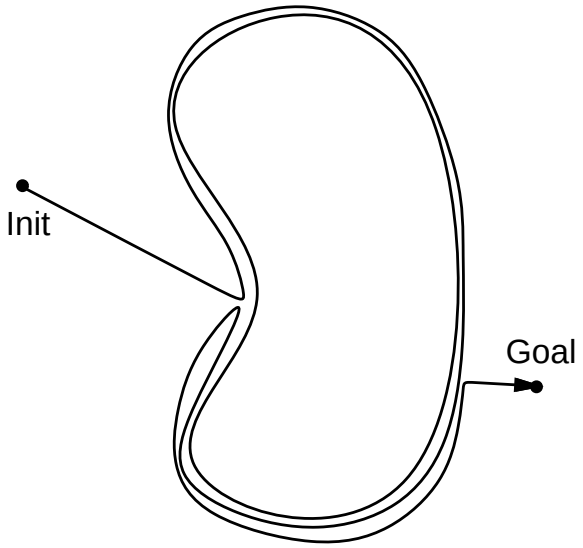
In the example, suppose that a train will arrive with probability $\frac{1}{4}$ (thus, there will not be a train with probability $\frac{3}{4}$). If the robot choose to go straight, the expected loss is $1(\frac{3}{4}) + 4(\frac{1}{4}) = 1.75$. If the robot takes the longer route, the expected loss is clearly 3. In this case, the robot should go straight. If the probability that a train would arrive is higher, such as $\frac{3}{4}$, then the expected-case analysis would lead the robot to select the longer route, as one would intuitively expect.

10.2 BUG Algorithms

This section addresses a motion strategy problem that deals with uncertainty with sensing. The BUG algorithms make the following assumptions:

- The robot is a point in a 2D world.
- The obstacles are unknown and nonconvex.
- An initial and goal positions are defined.
- The robot is equipped with a short-range sensor that can detect an obstacle boundary from a very short distance. This allows the robot to execute a trajectory that follows the obstacle boundary.
- The robot has a sensor that allows it to always know the direction and Euclidean distance to the goal.

BUG1 This robot moves in the direction of the goal until an obstacle is encountered. A canonical direction is followed (clockwise) until the location of the initial encounter is reached. The robot then follows the boundary to reach the point along the boundary that is closest to the goal. At this location, the robot moves directly toward the goal. If another obstacle is encountered, the same procedure is applied.

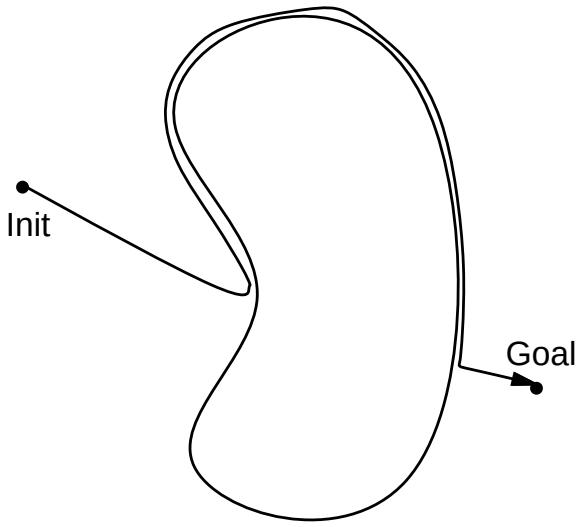


The worst case performance, L, is

$$L \leq d + \frac{3}{2}\sum_{i=1}^{N} p_i$$

in which d is the Euclidean distance from the initial position to the goal position, p_i is the perimeter of the i^{th} obstacle, and N is the number of obstacles.

BUG2 In this algorithm, the robot always attempts to move along the line of sight toward the goal. If an obstacle is encountered, a canonical direction is followed until the line of sight is encountered.



The worst case performance, L, is

$$L \leq d + \frac{1}{2}\sum_{i=1}^{N} n_i p_i$$

in which n_i is the number of times the i^{th} obstacle crosses the line segment between the initial position and goal position.

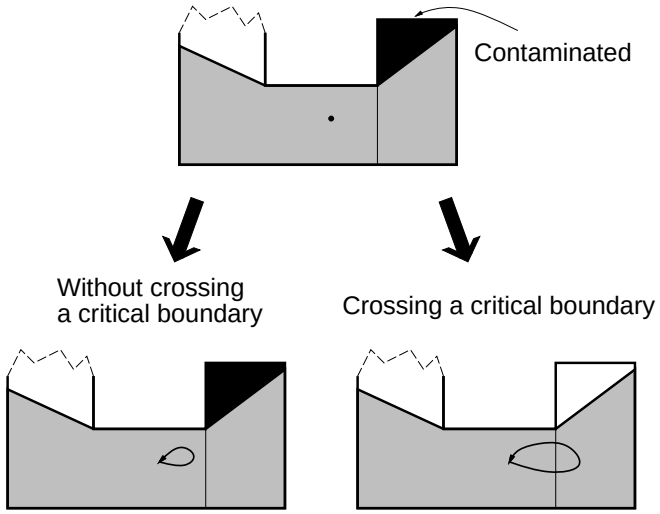
10.3 Visibility-Based Pursuit-Evasion

This section addresses another motion strategy problem that deals with uncertainty in sensing. The model is:

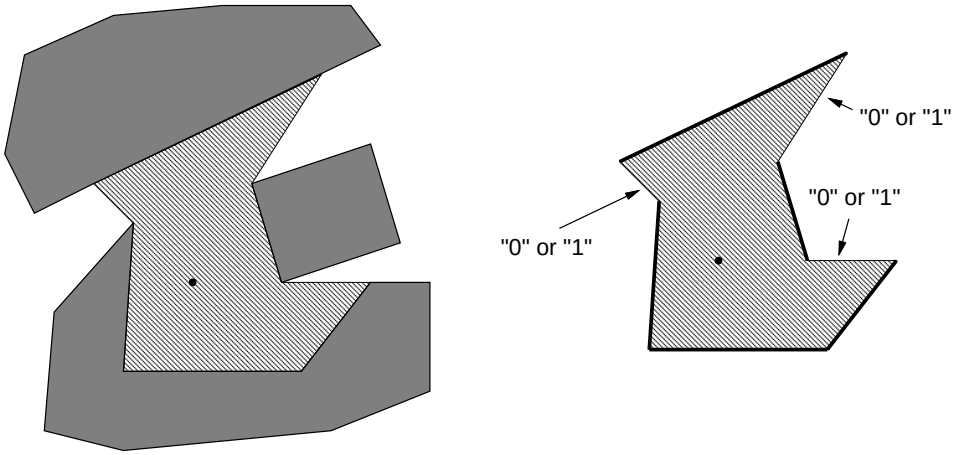
- The robot is a point (the "pursuer") that moves in a 2D world that is bounded by a simple polygon (it is simply-connected).
- The world contains a point "evader" that can move arbitrarily fast.
- The task is to move the pursuer along a path that guarantees that the evader will eventually be seen using line-of-sight visibility in the 2D world.

The problem can be formulated as a search in an *information space*, in which each information state is of the form (q, S). The information state represents the position of the pursuer, q, and the set, S, of places where the evader could be hiding.

The key idea in developing a complete algorithm that will construct a solution if one exists is to partition the world into cells, such that inside of each cell there are no critical changes in information.



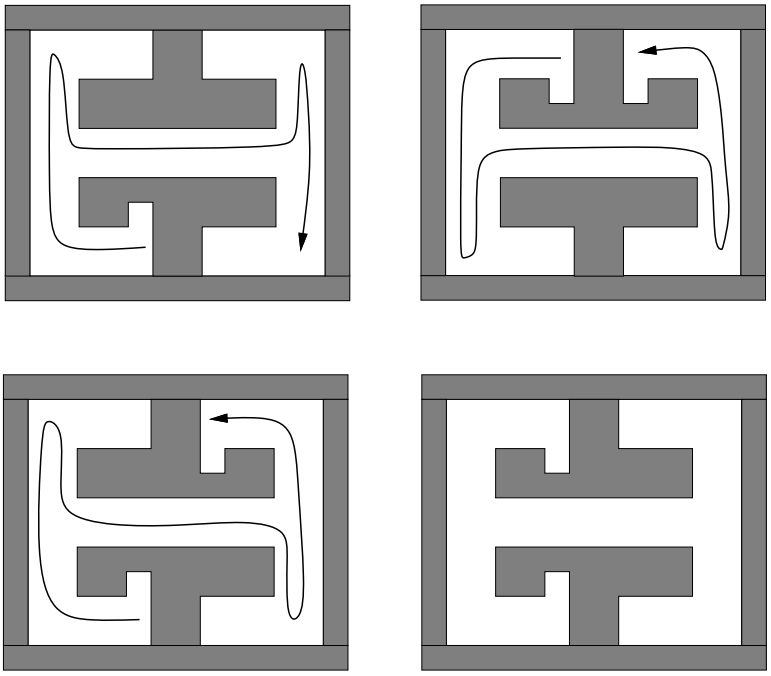
A finite graph search can be performed over these cells, cells might generally be visited multiple times. As the pursuer moves from cell to cell, the information state is maintained by maintaining binary labels on the gaps in visibility.



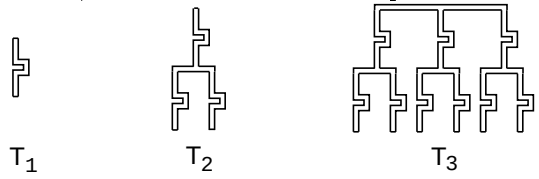
As the pursuer moves, gaps can generally split, merge, appear, or disappear, but within a cell, none of these changes occur. When a transition occurs from one cell to another, a simple transition rule specifies the new information state.

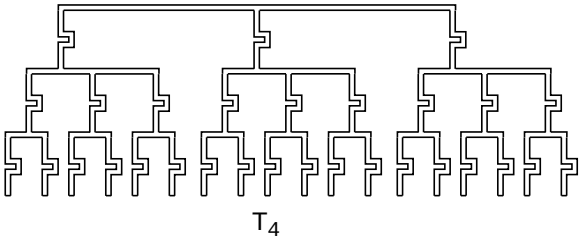
Examples

Even though there are slight variations in the environment from example to example, all of these can be solved, except for the last one.

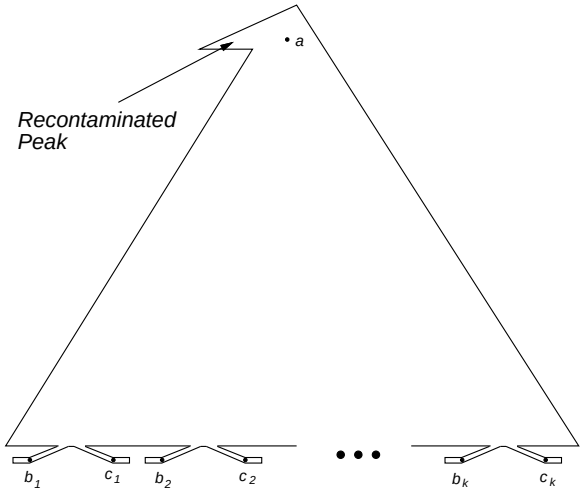


Each example below is labeled T_i, in which i is the number pursuers needed to solve the problem.





This example requires the peak to be visited $k-1$ times for k pairs of "feet".



Bibliography