A POMDP for Optimal Motion Planning with Uncertain Dynamics

Nicolas Meuleau[*] and **Christian Plaunt** and **David E. Smith** and **Tristan Smith**[†]
Intelligent Systems Division
NASA Ames Research Center
Moffet Field, California 94035-0001
{nicolas.f.meuleau, christian.j.plaunt, david.smith, tristan.b.smith}@nasa.gov

Abstract

This paper describes an approach to the problem of optimal motion planning with uncertain dynamics. This problem can occur whenever a vehicle suffers damage or when the environment makes the effect of motion actions unpredictable and potentially risky. We address in particular the case of aircraft with structural and/or control surface damage. The goal in this problem is to find an optimal plan for emergency landing, which might entail additional exploration of the aircraft flight envelope. However, this exploration is risky, and must be balanced by possible improvements in the resulting emergency landing plan. Evaluating the risk and potential benefit of exploration allows focusing on the fraction of the envelope that is beneficial, given the current situation (obstacles and possible landing sites). This reduces both the risk and the cost of exploration. The paper surveys previous work on optimal motion planning and flight envelope exploration. It shows how the problem of interleaving both tasks can be modeled as a Partially Observable Markov Decision Process where local dependencies between control states are modeled using Markov Random Fields.

Introduction

Motion Planning is the task of determining a sequence of movements that brings a robot and its environment to a desired state. The challenges inherent to this domain include dealing with the continuous nature of the state space, checking and avoiding collisions with obstacles, and accommodating the robots dynamics. Extensive previous work has brought efficient solutions to these issues, and impressive applications of motion planning have been developed. See Choset et al. (2004) for a survey of the field.

Robotic vehicle dynamics is a set of constraints on the trajectories it can follow. A constraint is *non-holonomic* if it cannot be expressed in terms of the robot position only, but it also involves the robot velocities. For instance, a static robotic arm with no inertia does not obey any such constraint. At each instant, the robot's movement can be reversed to bring it back in its initial position. Conversely, a drone is strongly bounded by non-holonomic constraints, and it takes skills to fly it without stalling or crashing.

In this paper, we focus on a problem in motion planning that has not been addressed yet (to our knowledge): planning for a robot that is strongly bounded by non-holonomic constraints *which are not known with certainty*. The robot's dynamics must be explored through risky *sensory* or *exploratory* actions. Subsequent motion planning depends on the results of these actions and aims at producing trajectories that are optimal or near-optimal for some cost function.

Our motivation comes from the Integrated Resilient Aircraft Control (IRAC) project founded by the NASA Aviation Safety Program. As part of this project, we address the problem of (automatically) planning emergency landing routes for aircraft that has suffered possibly severe structural and/or control surface damage, and whose maneuverability is largely uncertain. Observing the plane's reaction to small control inputs allows exploring the *flight envelope*, incurring the risk of losing control of the plane. This is however necessary to determine which moves are doable and which are not. Once sufficient envelope information is gathered, a least risky trajectory consistent with this info must be planned.

Previous work addresses both the problem of exploring the flight envelope independently of planning a route (Yi and Atkins 2010), and the problem of emergency landing planning when the dynamics are known with certainty (Meuleau et al. 2009). In this paper, we consider interleaving both sub-tasks. The idea is to limit risky exploration of the flight envelope to only what is necessary given the plane's situation (obstacles and possible landing sites). The basic technical challenge is to assess and balance the risk and benefit of every sensory action. The risk is to lose control of the plane, and the benefit is allowing new landing routes. Our goal is to produce *contingent flight plans*, where the route followed depends on the results of exploratory actions.

The technical approach adopted in this paper is to model the problem as a partially observable Markov decision process (POMDP) (Kaelbling, Littman, and Cassandra 1998). This is consistent with previous work on emergency landing planning with known dynamics, which is modelled as a fully observable MDP. Bayesian inference as used in POMDP theory is the most advanced technique to handle unobservability in a robot environment (Thrun, Burgard, and Fox 2005). Therefore, it is straightforward to consider this

[*]Carnegie Mellon University
[†]Mission Critical Technologies
ICAPS 2010 POMDP Practitioners Workshop, May 12, 2010, Toronto, Canada.

approach for modelling uncertainty in the dynamics. Of course, optimal Bayesian planning comes at a cost. The POMDP model is well known to be intractable, and it is a challenge to apply it to a real world application such as emergency landing planning. Our success relies on developing POMDP models and algorithms that capture the most important features of the problem while remaining tractable. This paper presents our propositions to achieve this goal.

The paper is organized as follows: We first outline previous work on flight envelope exploration. Next, we show how the uncertainty on the flight envelope can be modelled as a particular probability distribution known as a Markov Random Field (MRF). Then we survey previous work on emergency landing planning with known dynamics, that is an instance of MDP planning. We show how this work can be extended into a POMDP model that accounts for the uncertainty on the flight envelope. Finally, we survey a preliminary implementation of the model.

Flight Envelope Exploration

Our approach to flight envelope exploration builds on previous work by Atkins and colleagues (Tang, Atkins, and Sanner 2007; Yi and Atkins 2010) that represents the state-of-the-art on the topic. This work is based on the idea that the plane must fly only through *stable trim states*. A stable trim state is a equilibrium where all plane velocities stay constant as long as control input does not change. Atkins et al. propose a way to model the process of moving from a known stable trim state to a new state initially uncertain.

Formally, a *control configuration* z is defined as a vector containing the plane altitude, air speed, climb rate and turn rate.[1] That is four continuous variables whose values are bounded. A *control input* μ is a four-dimensional vector containing throttle and flight surfaces inputs. *Control states* are defined as pairs $x = (z, \mu)$. The *dynamics* of the plane is a quadratic function $f(z, \mu)$ such that, at each time t: $\dot{z} = \partial z/\partial t = f(z, \mu)$. A stable trim state is then defined as a pair (z, μ) such that $f(z, \mu) = 0$. In a stable trim state (z, μ), the configuration z is constant as long as the input μ is constant. The set of all stable trim states is denoted 0_f. The model also includes a notion of *stabilizable trim state*. It represents a control state x that can be held only with the help of the adaptive controller. Control states that are neither stable nor stabilizable trim states are called *unstable*. In this work, we define the *flight envelope* ρ as the set of all stable and stabilizable trim states.

Atkins assumes that, following the incident that triggered the emergency, an adaptive controller has brought the plane to an initial stable trim state. Then they proceed to exploring the control state space step by step. All state variables are discretized regularly, so that the space to explore is a multidimensional grid where contiguous states are connected. Each grid node takes one of the three values *stable*, *stabilizable* or *unstable*.

At each time t, a linear approximation of f called Δ is computed at the current state, using the adaptive controller. This linear approximation is used to estimate the stability of neighboring states. The stable states are approximated by the set 0_Δ containing all states where Δ is zero. In most cases, it intersects 0_f only at the current state. That is, all other states of 0_Δ are mistakenly estimated as stable. Nevertheless, exploration is performed by attempting to move to a state $x' = (z', \mu') \in 0_\Delta$. As this state is not really stable, a control correction δ_μ is attempted by the adaptive controller to bring the system to a stable trim state. This process is uncertain and has three possible outcomes:

1. Everything works as expected and the system makes a safe transition to the stable trim state $(z', \mu' + \delta_\mu)$ where a new linear approximation is computed;
2. The control correction δ_μ is too large and we loose control of the plane before reaching a new trim state. This may also happen because there is no control correction δ_μ that can make $(z', \mu' + \delta_\mu)$ a trim state. In both cases, we fail and the plane crashes;
3. The adaptive controller is only able to reach a stabilizable trim state $(z', \mu' + \delta_\mu)$. This is a warning sign indicating that we are close to the edge of the flight envelope. No further exploration should be performed in that direction.

The larger the step we attempt, the bigger the probability of failing or ending up in a stabilizable state.

Atkins et al. propose an efficient way to model the process of exploring the flight envelope. To decide which move to perform in the configuration space, they use a set of predefined exploration policies that follow the edges of the flight envelope using bug algorithms, and hand-crafted rules to decide which policy to use in a given situation. Overall, exploration of the flight envelope can represent up to 20 minutes of flight time, most of it being spent waiting for the adaptive controller to stabilize the plane. In this paper, propose to trade part of this execution time for computation time. Our approach is to limit the amount of flight envelope exploration by coupling it with the problem of optimal path planning. We want to assess the relevance of an exploration move before performing it. Such reasoning must include information about the plane's situation that is not taken into account in Atkins's work, including possible landing sites and their characteristics, weather and landscape obstacles. In a sense, our goal may be seen as to automatically decide which exploration policy to use in each state encountered, and to automatically design new exploration policies.

Modeling Uncertainty

Approaching flight envelope exploration as a POMDP requires representing the uncertainty as a *belief* on the flight envelope, that is, a probability distribution $b(\rho)$ on the unknown parameter ρ. This section describes how we address this important issue.

Following Atkins work, we do not assume a centralized model of the uncertain dynamics f, but rather a collection

[1] Our terminology and definitions vary slightly from Atkins et al. In fact, our choices of vocabulary probably collide with traditional control theory terminology. However, this work borrows from several areas of computer science that use different notions of state and configuration. Therefore, it appears important to us to avoid all ambiguities and to maintain a certain coherence in the terminology by using a "private" set of definitions, at the cost of being in contradiction with part of the literature.

of random variables distributed at the nodes of a discrete grid in the control state space. We assume that each discrete state x takes one of the three values stable, stabilizable or unstable, and that we can observe with certainty the value of a control state when we visit it. In Atkins work, we also observe the linear approximation of the dynamics Δ computed in each visited state. It is the collection of partial derivatives of f with respect to all control state variables. It can be used to estimate the probability of success of a move in every direction. In a sense, Δ represents noisy information about the value of each neighboring state. This observation introduces an implicit dependency between two neighboring control states x and x':

$$\Pr(\omega_{x'} \mid \omega_x) = \sum_{\Delta} \Pr(\omega_{x'} \mid \Delta) \Pr(\Delta \mid \omega_x) \ .$$

where ω_x is an assignment of value to state x.

Although it does not feature explicit observations of each neighbor, our model of uncertainty does include dependencies between neighboring states ($\Pr(\omega'_x \mid \omega_x)$). These dependencies appear to us as a fundamental feature of the problem that a realistic model should not neglect. In our simple three-valued model, they are sufficient to generate rational behavior: if we observe that a state is stabilizable, we can directly infer that there is a high risk that its uncertain neighbors are unstable. Conversely, neglecting these dependencies can lead to dramatic policies. Suppose, for instance, that the system has estimated that it is worthwhile exploring right bank up to $40\,^\circ$ (all other variables being equal), but encounters an unstable state at $20\,^\circ$ of bank. If it assumes independence between neighboring states, it is unable to propagate the bad news about $20\,^\circ$ bank to other control states. So, it still estimates that it is worthwhile pushing exploration towards $40\,^\circ$ of bank, and then crashes the plane.

Taking into account dependencies between neighboring states is also important because it allows generalizing the value of visited states to uncertain states. Generalization is a classical capability of machine learning algorithms (Mitchell 1997). In the case of emergency landing, it allows inferring that a state is very probably flyable/unstable if a sufficient number of states in its vicinity are known to be flyable/unstable with certainty. It helps minimizing the number of exploration moves. This is illustrated in Fig. 1.

The next step in building the model is to look at the dependencies between states that are not neighbours in the control state grid. We observe a form of Markov property exhibited in Atkins et al. 's model. Suppose again that we are exploring right bank, all other variables being constant. At time $t-2$ with $18\,^\circ$ bank, we can estimate the probability that $20\,^\circ$ bank is flyable based on the linear approximation of f at the current state. At $t-1$, with $19\,^\circ$, bank we can build a new estimate of $20\,^\circ$ bank that is more accurate, because we are closer to this state. In fact, the new estimate totally overrules the old one: the (estimated) probability of the outcome at time t with $20\,^\circ$ bank depends only on the outcome at time $t-1$, and not on previous time steps. We recognize here the Markov property that is familiar to the MDP community. In fact, Markov property is more appropriately expressed spatially rather than temporally: the probability of a state value

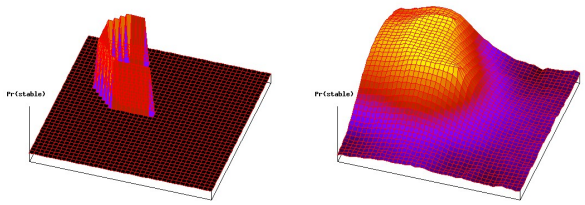

Figure 1: Generalization in Markov Random Fields: Starting from a belief giving uniformly $\Pr(\text{stable}) = 0.5$ in all $x \in X$, we have observed that states are stable with certainty along a trajectory. *Left:* We assume independence between neighboring states. *Right:* Assuming dependencies between neighbors allows generalizing the value of known states to states in their vicinity.

depends only on its neighbors values :

$$\Pr(\omega_x \mid \omega_{x'}, x' \neq x) = \Pr(\omega_x \mid \omega_{x'}, x' \in N(x)) \ , \quad (1)$$

where $N(x)$ is the set of neighbors of x in the grid.

Equation 1 is characteristic of a class of probability distributions widely used in computer vision and known as Markov Random Fields (MRFs) (Kindermann and Snell 1980). Using notations compatible with our application, an MRF is defined over a graph $\mathcal{G} = (X, N)$ where X is a set of vertexes (control states) and N is a neighborhood function implicitly representing a set of edges (the edges joining adjacent control states). Assuming that each node x can take a random value over a finite set ({stable, stabilizable, unstable}), an MRF is defined as a joint probability distribution over the values of all $x \in X$ that satisfies Eqn. 1.

A fundamental result in the study of MRFs is the so called *Markov-Gibbs equivalence*. First, we remind that a *clique* of $\mathcal{G}$ is a subset of nodes $C \subseteq X$ such that every pair of states in C are neighbors. (Singletons $\{x\}$ are also considered as cliques.). Next, a *map* ω is defined as an assignment of value to all control states $x \in X$.[2] We use ω_C to represent the map ω restricted to clique C. We can think of ω_C as an assignment of value to all states $x \in C$. Then:

- A *(nearest-neighbor) Gibbs potential* $\mathcal{V}$ is a function that assigns a real value $\mathcal{V}_C(\omega_C)$ to every map ω_C of every clique C in $\mathcal{G}$;
- Given a Gibbs potential, the *energy* $U(\omega)$ is defined on the set of maps as: $U(\omega) = -\sum_C \mathcal{V}_C(\omega_C)$;
- The *Gibbs measure* (induced by U) is the joint probability distribution on X defined by:

$$\Pr(\omega) = \frac{e^{U(\omega)}}{\sum_{\omega'} e^{U(\omega')}} = \frac{e^{-\sum_C \mathcal{V}_C(\omega_C)}}{\sum_{\omega'} e^{-\sum_C \mathcal{V}_C(\omega'_C)}} \ . \quad (2)$$

The fundamental result of MRF theory is that, given a graph $\mathcal{G} = (X, N)$, a joint probability distribution on X is an MRF (i.e, satisfies Eqn.1) *if and only if* it is a Gibbs measure.

The Markov-Gibbs equivalence shows that an MRF belief distribution $b(\rho)$ is completely represented by the set of

[2] This is again a personal terminology. The term used by mathematicians to designate a *map* as defined here is *configuration* !

potentials $\mathcal{V}_C(\omega_C)$ for all cliques C. In the control state grid, the only cliques are the singletons $\{x\}$ and the pairs (x, x'). Therefore, a belief is totally defined by: (i) The unary potential $\mathcal{V}_x(\omega_x)$ for all $x \in X$; (ii) The binary potential $\mathcal{V}_{x,x'}(\omega_x, \omega_{x'})$ for all $(x, x') \in X^2$. Binary potentials is cumbersome data that can consume lots of memory space. For this reason, we do not encode them by defining an explicit value for each pair (x, x'). Instead, we use a compact encoding that allows computing all binary potentials based on a minimum information. For instance, a local pattern can be defined and applied in all states. In the right graph of Fig. 1, a simple *diffusion* process is repeated identically over the whole grid: Assigning a value of a state uniformly pushes its neighbors to take the same value.

The conditionals probabilities of an MRF can conveniently be computed using:

$$\Pr(\omega_x \mid \omega_{x'}, x' \in N(x)) = \frac{\exp^{-\sum_{C:x\in C} V_c(\omega_x, \omega_{x'})}}{Z} \ , \quad (3)$$

where Z is a normalizing constant. However, this compact formula applies only to the case where the value of *all* neighbors of x is known. When solving the emergency landing planning POMDP, we estimate the risk of exploring a state x by the probability that x is unstable given the current belief. In most cases, we do not know the value of all neighbors of x when we perform this computation. Sometimes, we might not even know the value of any of them (if we allow large jumps in the control state space). In fact, rather than the conditional probabilities of Eqn. 3, we need the marginal probabilities $\Pr(\omega_x \mid b)$ for all x and all ω_x. Unfortunately, computing marginals in an MRF is known to be intractable, but there are efficient approximation techniques available. In this work, we use the Markov Chain Monte Carlo (MCMC) technique known as Gibbs sampling to compute marginal probabilities (Casella and George 1992). The right graph of Fig. 1 was generated using this technique.

Another important issue in POMDP models is computing posterior beliefs. If b is the current belief and we observe o, then the posterior belief $\mathcal{B}(b, o)$ is given by Bayes' rule:

$$\mathcal{B}(b, o)(\rho) = \frac{b(\rho)\Pr(o \mid b, \rho)}{\sum_{\rho'} b(\rho')\Pr(o \mid b, \rho')} \ . \quad (4)$$

In the case of our application, this computation is largely simplified by the fact that there is no noise in observations: when we visit a state, we observe the value of this state with certainty. (Moreover, as explained before, our model does not feature an explicit noisy observation of neighboring states.) The MRF representing the belief $\mathcal{B}(b, \bar{\omega}_x)$ posterior to observing that state x has value $\bar{\omega}_x$ is simply obtained by fixing x to $\bar{\omega}_x$ in the MRF b, that is:

- The unary potentials $\mathcal{V}_{x'}(\omega_{x'})$ of all states x' such that $x' \neq x$ and $x' \notin N(x)$ are unchanged;
- The binary potentials $\mathcal{V}_{x',x''}(\omega_{x'}, \omega_{x''})$ of all state pairs (x', x'') such that $x' \neq x$ and $x'' \neq x$ are unchanged;
- For each neighbor x' of x, the unary potentials $\mathcal{V}_{x'}(\omega_{x'})$ are all augmented with the value $\mathcal{V}_{x,x'}(\bar{\omega}_x, \omega_{x'})$;
- The unary potentials $\mathcal{V}_x(\omega_x)$ as well as all binary potentials involving x are discarded. x is not a random variable of the MRF anymore. It can be pruned from the graph $\mathcal{G}$.

In fact, it is not necessary to compute these potentials explicitly for each observation $\bar{\omega}_x$. Gibbs sampling can estimate the marginals of $\mathcal{B}(b, \bar{\omega}_x)$ using the potentials of the prior belief b. In counterpart, every sample of a value for x is assumed to return the value $\bar{\omega}_x$. To represent an initial belief and a set of posteriors derived by observing the value of one or several states, we need to store only one copy of the unary and binary potentials. Each belief is explicitly represented by the set of known state values, and the Gibbs sampler uses this information directly to estimate all marginals.

Emergency Landing Planning

An important previous achievement of the IRAC project is a system able to recommend landing sites and trajectories based on two pieces of information: the plane's flight envelope and its environment. The later includes the landscape configuration, weather obstacles, and a list of possible landing sites with fine estimates of the conditions at each site (Meuleau et al. 2009). The problem addressed here is a more classical motion planning problem in the sense that the robot dynamics are known with certainty. However, it exhibits a series of difficulties that, although all addressed by previous literature (Choset et al. 2004), are rarely put together in the same application:

- Although the dynamics are known with certainty, they constrain strongly the set of possible trajectories. An aircraft is arguably among the most non-holonomic systems that needs to be auto-piloted;
- Conversely to most motion planning applications, we are not interested in finding any possible trajectory, but we want a solution that minimizes the risk as much as possible. In other words, this is a problem of *optimal* motion planning where the cost function is the failure probability;
- There is not a single predefined goal state, but we must select the landing site that is the safest to reach;
- Some obstacles are soft: they can be traversed incurring a cost. For instance, we may fly through a zone of turbulence, but the probability of failing is higher.

Meuleau et al. (2009) presented the results of previous research on emergency landing planning with known plane dynamics, and experiments on real-size instances of the problem. Some of their best results are obtained using a variant of the probabilistic roadmap (PRM) algorithm called DVPRM by the authors. This is not surprising, given that probabilistic algorithms have become the approach of choice in many robotic applications, particularly in the presence of dynamics. The PRM algorithm builds a discrete roadmap in a continuous space by drawing waypoints at random, and connecting each waypoint to a fixed number of nearest neighbors, as long as these connections do not traverse any obstacle. Waypoints representing the start and goal locations are added and connected to the roadmap. Then a search algorithm is used to find a path between these two waypoints. DVPRM is a variant of the PRM approach adapted to the specificity of the emergency landing problem.

Waypoints: DVPRM uses the notion of *robot configuration* q that includes the plane latitude, longitude, altitude,

heading and speed. Note the difference with the control configuration z defined above. The control configuration does not include the plane position (latitude and longitude) and heading. This is because all variables with which control is concerned (mostly linear and angular velocities) evolve independently of where the plane is located on the map and where it is heading. (Altitude is important because it affects the air density.) Conversely, the robot configuration determines the exact position of the plane on the map. It is necessary if we want to take into account obstacles in the environment. In counterpart, the robot configuration excludes most velocity information. As explained below, these variables influence the cost of the edges of the roadmap.

Roadmap: Robot configurations are generated at random within the resource bounds of the plane. Then neighboring configurations are connected if no hard obstacle interferes. Edges can traverse soft obstacles but it impacts their cost. To account for the plane dynamics, edges are oriented and two edges with different orientation and cost are created for each pair of neighboring waypoints. A waypoint is added to the roadmap to represent the plane and each possible landing site.[3] Because it seeks a near-optimal solution, DVPRM allows cycles in the roadmap. In other words, the roadmap is not limited to a tree as in the popular Rapidly-expanding Random Tree (RRT) and its variants. By generating several solutions, we increase our chance to generate a good one.

The most original feature of the algorithm is to incorporate waypoints and edges from the *visibility graph* to the PRM. The visibility graph is an early approach to polygonal obstacle avoidance. It contains a node for each obstacle corner and for the start and goal locations. Two nodes are connected if the segment between them does not traverse any (hard) obstacle. In 2D, the visibility graph is guaranteed to contain an optimal path, but this does not generalize to higher dimensions and soft obstacles. Nevertheless, Meuleau et al. (2009) proposed to extend visibility graph nodes in several dimensions to make them well defined robot configurations. These waypoints are then added and connected to the probabilistic roadmap. They are ideal for skirting obstacles efficiently. They help the algorithm find shorter and smoother paths around obstacles.

Edge cost: The cost of an edge between two robot configurations q and q' is a failure probability that reflects the plane's limited maneuverability, the presence of obstacles in the environment, and limited resources. First, a *trajectory planner* (also developed in the IRAC project) is used to compute the shortest trajectory between q and q'. The trajectory planner takes as input the plane's flight envelope ρ and outputs a trajectory that stays within the maneuverability constraints. This is done by essentially computing Dubins path between the waypoints. In general, less constrained flight envelopes allow shorter and safer trajectories. The difference can be pretty dramatic. For instance, limiting left bank can prevent a short left turn and force a large and costly turn to the right. Next, we check if this trajectory exhausts any resource and set the failure probability to one if it happens. Finally, the trajectory is checked for obstacle collisions, and its failure probability is estimated. It is as a function of the total length of the trajectory, and the length travelled in each obstacle. Edge computation is the most expensive part of the algorithm. Therefore, DVPRM uses a lazy approach that does not compute the traversibility and cost of an edge until that edge is reached during search. Once computed, results are stored for the rest of the search.

Search: Search is performed by the standard A* algorithm. The heuristic value of a robot configuration is obtained by computing the Euclidean distance from its latitude and longitude to the closest goal's latitude and longitude, and assuming this distance is flight in clear weather. This admissible heuristic guarantees that optimal solutions are found, while allowing pruning a consequent portion of the search space. If there are resource constraints such as a maximum distance travelled and/or time before landing, A* search nodes must be augmented with a variable representing the remaining level of each limited resource. That is, two search nodes are considered equal if they represent the same robot configuration, and if they have the same level of remaining resources. This is necessary to guarantee that the sub-graph below a node depends only on this node, and not on the path to this node.

The POMDP

In this paper, we model the problem of interleaving flight envelope exploration and emergency landing planning as a Partially Observable Markov Decision Process (POMDP) (Kaelbling, Littman, and Cassandra 1998). This is consistent with previous work on planning with known dynamics, which may be seen as an instance of MDP planning. An *MDP-state* of the emergency landing problem is a triple $s = (q, \theta, \rho)$ where q is a robot configuration, θ represents the levels of remaining resources, and ρ is a representation of the flight envelope. In other words, the triple (q, θ, ρ) contains all the information necessary to determine the success probabilities of future moves. DVPRM actually finds the solution of the following Bellman equation:

$$V(\text{landed}, \theta, \rho) = 1 \ , \qquad V(\text{crashed}, \theta, \rho) = 0 \ ,$$
$$V(q, \theta, \rho) =$$
$$\max_{q' \in M(q)} \left[\Pr(\text{ succ } \mid q, \theta, \rho, q') V(q', \theta - C(q, \rho, q'), \rho) \right] \ ,$$

where q is a robot configuration where the plane is neither landed nor crashed, $M(q)$ is the set of neighbors of q in the roadmap, $\Pr(\text{ succ } \mid q, \theta, \rho, q')$ if the probability of successfully flying from q to q' given the resource θ and flight envelope ρ, and $C(q, \rho, q')$ is the cost of flying from q to q' in terms of resources. The solution is a set of values $V(s)$ representing the probability of successfully landing if we are in state s and follow an optimal flight plan, for each state s reachable by the optimal solution. Note that the flight envelope information ρ is constant throughout, and so it can be omitted from all equations. The new MDP-states $s = (q, \theta)$ are exactly the search nodes of the A* algorithm in DVPRM.

The MDP model is turned into a POMDP to account for flight envelope uncertainty in the following way. A

[3] Meuleau et al. (2009) used a new roadmap for each possible landing site. However, in the POMDP model that would not allow selecting the landing site based on the results of exploration moves.

POMDP-state is defined a s a triple $\tilde{s} = (q, \theta, b)$ where b is a *belief* on the unknown parameters, that is, a probability distribution $b(\rho)$. The new Bellman equation is:

$$\begin{aligned} &\tilde{V}(\text{landed}, \theta, b) = 1 \ , \quad \tilde{V}(\text{crashed}, \theta, b) = 0 \ , \\ &\tilde{V}(q, \theta, b) = \max\left\{\tilde{V}_1(q, \theta, b), \tilde{V}_2(q, \theta, b)\right\} \ , \end{aligned} \tag{5}$$

$$\tilde{V}_1(q, \theta, b) = \max_{q' \in M(q)} \left[\Pr(\text{ succ} \mid q, \theta, b, q')\tilde{V}(q', \theta - C(q, b, q'), b)\right] \ ,$$

$$\tilde{V}_2(q, \theta, b) = \max_a \left[\sum_o \Pr(o \mid q, b, a) \Pr(\text{ succ} \mid q, \theta, \mathcal{B}(b, o), S(q, a, o)) \tilde{V}\left(S(q, a, o), \theta - C(b, a, o), \mathcal{B}(b, o)\right)\right] .$$

Two Types of Actions: Equation 5 reads as follows: in every POMDP state $\tilde{s}$, we can chose from two types of action:
1. *Ordinary actions* are exploration-safe actions: they do not traverse any control state whose value is uncertain. They represent alternative $\tilde{V}_1$ in Bellman equation. Following the choice of destination waypoint q', we incur a risk of failing and a resource cost that depend on the current belief b:

$$\Pr(\text{ succ} \mid q, \theta, b, q') = \mathrm{E}_\rho\left[\Pr(\text{ succ} \mid q, \theta, \rho, q') \mid b\right] \ ,$$
$$C(q, b, q') = \mathrm{E}_\rho\left[C(q, \rho, q') \mid b\right] \ .$$

As ordinary actions do not provide any new information on the flight envelope, the belief is unchanged in the arrival state: $s' = (q', \theta - C(q, b, q'), b)$. In practice, the risk $\Pr(\text{ succ} \mid q, \theta, b, q')$ and cost $C(q, b, q')$ are computed by querying the trajectory planner using a (certain) model of the flight envelope $\bar{\rho}(b)$ defined as:

$$\bar{\rho}(b) = \{x : \Pr(x \neq \text{unstable} \mid b) \geq 1 - \epsilon_{\text{safe}}\} \ . \tag{6}$$

That is, $\bar{\rho}(b)$ contains all control states x that have a probability of being flyable greater or equal to $1 - \epsilon_{\text{safe}}$, where $\epsilon_{\text{safe}} \simeq 0$ is an algorithm parameter. The cost reflects the risk associated with flying from q to q' given the current conditions. It neglects the risk associated with the uncertain flight envelope, which is bounded by ϵ_{safe} in each control state.
2. *Exploratory (or sensory) actions* do not aim at advancing the plane toward a goal. Instead, their purpose is to explore uncertain control states. They represent option $\tilde{V}_2$ in Bellman equation. An exploratory action a is modelled as a trajectory in the (discretized) control state space from a reference state $x_0(q)$ to a state with uncertain value, and then back to $x_0(q)$. (x_0 depends on q because control states and robot configurations are bounded by the altitude and speed variables.) This exploration provides an observation o about the value of some control configurations, according to the probability distribution $\Pr(o \mid q, b, a)$. An exploration move may encounter an unstable state and cause a plane crash, which is modeled by observation o_{fail}. If successful, it leads to robot configuration $S(q, a, o)$ (S stands for "successor"), with a resource cost $C(b, a, o)$. It is convenient to have the final configuration S and cost C depend on the action outcome o. Then we can model a crash during exploration by setting $S(q, a, o_{\text{fail}}) = \text{crashed}$. The new belief $\mathcal{B}(b, o)$ is obtained by applying Bayes' rule as in Eqn. 4.

The risk associated with flying from q to its successor while executing a is represented by the factor $\Pr(\text{ succ} \mid q, \theta, \mathcal{B}(b, o), S(q, a, o))$. This probability is computed as in the equation of $\tilde{V}_1$, by passing the certain model of the flight envelope $\bar{\rho}(\mathcal{B}(b, o))$ defined by Eqn. 6 to the trajectory planner, then checking for obstacle traversal. Note that we use the posterior belief instead of the current belief to compute $\bar{\rho}$. It reflects the fact that the exploration action successfully traverses control states appearing uncertain according to b.

In the rest of this paper, we assume that $S(q, a, o) = q$ for all $o \neq o_{\text{fail}}$. In other words, we assume that the effects of exploration moves are negligible at the scale of path planning (as long as the plane does not crash). To simplify notation, we also assume that $C(b, a, o) = 0$ for all b, a, o. The equation of $\tilde{V}_2$ becomes simply:

$$\tilde{V}_2(q, \theta, b) = \max_a \left[\sum_{o \neq o_{\text{fail}}} \Pr(o \mid q, b, a)\tilde{V}\left(q, \theta, \mathcal{B}(b, o)\right)\right] .$$

The cost of an exploration move reflects only the risk due to the uncertainty in the flight envelope, through the term $\Pr(o_{\text{fail}} \mid q, b, a) = 1 - \sum_{o \neq o_{\text{fail}}} \Pr(o \mid q, b, a)$. It neglects the risk due to flying in the environment during exploration.

Layered Roadmap: Sensory actions may have different outcomes o, and the optimal POMDP policy may follow a different course of actions for each outcome. So, exploratory actions introduce branches in the plan, as many branches as there are possible outcomes $o \neq o_{\text{fail}}$. (We do not need to plan for contingency o_{fail} because this observation interrupts plan execution on a failure.) Ordinary actions have two possible outcomes, one being plan failure and the other a successful transition to a known successor waypoint. Therefore, they do not introduce any branch in the plan.

The POMDP can be represented as set of roadmaps $\mathcal{R}(b)$, one for each reachable belief b. These roadmaps use robot configurations q as waypoints, as in the fully observable case. (Resources are omitted and taken into account only by the search algorithm.) The system starts in $\mathcal{R}(b_0)$ for some initial belief b_0. Ordinary actions move the system inside of the current roadmap $\mathcal{R}(b)$ as they do not change the belief. Exploratory actions trigger a jump to a new roadmap $\mathcal{R}(\mathcal{B}(q, b, o))$ depending on the observation performed o. We call this structure the *layered roadmap*.

Our approach is based on representing each "layer" $\mathcal{R}(b)$ using a DVPRM roadmap. As generating and connecting waypoints is an expensive stage of DVPRM, we use the same waypoints and edges for each belief b. Only the cost of edges, computed using $\bar{\rho}(b)$, varies from one roadmap to the other. The whole POMDP is thus represented as a single roadmap where edges have multiple costs associated to different beliefs under which they are reachable.

Search and Heuristics: Search is performed using the AO* algorithm that is the variant of A* for AND-OR graphs

(graph with contingent branches). AO* search nodes are POMDP-states (q, θ, b) (they account for limited resources).

In the case of known dynamics, the heuristic value of an MDP-sate (q, θ, ρ) is the cost of flying in clear weather the Euclidean distance to the closest landing site. We denote it $h_{\mathrm{Euclid}}(q, \theta)$. Note that it does not depend on ρ, because it supposes a straight trajectory that does not respect plane dynamics. It is admissible, as it assumes that an optimistic distance estimate is flight in optimistic external conditions.

The heuristic h_{Euclid} can be used in the POMDP as well. However, it is quite inefficient because it forces the search to re-discover some features of the environment, such as obstacles, in each roadmap $\mathcal{R}(b)$. An efficient heuristic must be able to generalize observations made in a roadmap $\mathcal{R}(b)$ to other roadmaps $\mathcal{R}(b')$, $b' \neq b$. For instance, if there is a line of severe thunderstorms on the left of the plane, we should not explore the space on the left *in every roadmap*.

The POMDP heuristic $\tilde{h}_{\mathrm{optimist}}(q, \theta)$ exhibits the generalization capability. It is based on solving the emergency landing MDP using the flight envelope model $\bar{\bar{\rho}}$ defined as

$$\bar{\bar{\rho}}(b_0) = \{x : \Pr(x \neq \text{unstable} \mid b_0) > \epsilon_{\mathrm{doable}}\} \ ,$$

where $\epsilon_{\mathrm{doable}} \simeq 0$ is an algorithm parameter. That is, all uncertain states according to the initial belief b_0 are assumed flyable in this model. The heuristic $\tilde{h}_{\mathrm{optimist}}$ is then defined as the optimal solution of this MDP:

$$\tilde{h}_{\mathrm{optimist}}(q, \theta) = V(q, \theta, \bar{\bar{\rho}}(b_0)) \ . \tag{7}$$

It is an admissible heuristic because it uses an optimistic model of the flight envelope. Conversely to h_{Euclid}, it carries information about the plane environment across roadmaps. For instance, if a configuration q is assigned bad MDP-value because it is in the middle of costly obstacles, then search avoids it in all roadmaps $\mathcal{R}(b)$ of the POMDP.

Unfortunately, Eqn. 7 is difficult to implement exactly. The optimistic MDP is solved using DVPRM with the heuristic h_{Euclid} and the same set of waypoints and edges as in the POMDP layered PRM. It provides the optimal value $V(q, \theta, \bar{\bar{\rho}}(b_0))$ for all (q, θ) traversed by the optimal solution, but the values of other states are not guaranteed to be accurate. However, solving the POMDP requires the value $\tilde{h}_{\mathrm{optimist}}(q, \theta)$ for some pairs (q, θ) that do not belong to the MDP optimal solution. This may happen because configuration q is not traversed by the optimal solution. In this case, an accurate value for $V(q, \theta, \bar{\bar{\rho}}(b_0))$ can be computed by re-starting the MDP A* search from this state. In other cases, q is traversed by the optimal MDP policy, but with a different level of resource θ. In fact, the resource left at a given waypoint q almost always differ from a POMDP layer $\mathcal{R}(b)$ to the optimistic MDP roadmap. This is because a different model of the flight envelope is used to compute edge costs in each roadmap ($\bar{\rho}(b) \neq \bar{\bar{\rho}}(b_0)$). We cannot restart the A* search each time, because that would mean re-solving the optimistic MDP nearly for each state encountered by the POMDP AO* search. Therefore, we use the following scheme: When we need the heuristic value of a pair (q, θ), we check in the optimistic MDP whether there exists a resource level θ' such that (i) $V(q, \theta', \bar{\bar{\rho}}(b_0))$ is known with certainty, (ii) θ' represents a higher or same level of resource than θ, for each resource. If such θ' exists, we set $\tilde{h}_{\mathrm{optimist}}(q, \theta) = V(q, \theta', \bar{\bar{\rho}}(b_0))$. Otherwise, the MDP A* search is restarted from (q, θ). (If several θ' satisfy the conditions, we select the closest to θ.) Because higher levels of resources allow better solutions, this heuristic is admissible.

Macro-actions: Despite smart heuristics, we still face an exponentially large belief state space. Indeed, if we allow exploring each control state individually, then the number of belief states reachable is exponential in the side of the grid. In our application, this number is $3^{|X|}$ where X is the number of control states, because each state can be either unknown, stable or stabilizable according to the belief. (Again, there is no need to represent a belief where a state is unstable, because we crash when reaching it.) Therefore, we limit exploration actions to a finite set of exploration *macro-actions* (Hauskrecht et al. 1998). A macro-action A is defined as a trajectory $\mathcal{T}_A$ in the control state space from the reference state $x_0(q)$ to a target state x_A whose value is uncertain. Executing the macro-action consists of following $\mathcal{T}_A$ as long as the states traversed turn out to be stable. Exploration of trajectory $\mathcal{T}_A$ terminates: (i) when we reach the target x_A having encountered only stable states on the way; (ii) when we reach a stabilizable state, in which case we purposely interrupt exploration; (iii) if the plane crashes because we have traversed an unstable state. (In the first two cases, A also includes flying back to $x_0(q)$.)

An exploration action represented by a trajectory of length l has $l + 1$ possible outcomes that are worth planning for: observing that one of the states traversed is stabilizable ($l - 1$ outcomes); and going all the way to the target and observing that it is stable or stabilizable (2 outcomes). It contrast strongly with the 2^{l-1} outcomes that are possible if we explore all states traversed by $\mathcal{T}_A$ one by one. In fact, macro-actions save considerable computation time by implementing a strong heuristic: we must stop exploring a direction when we encounter a stabilizable state. An algorithm using only primitive actions defined as simple jumps in the control space must re-discover this principle itself, which implies high computational costs.

We further restrict the set of reachable beliefs by limiting exploration to a few number of macro-actions in each trajectory. As a consequence, the set of control states whose value can be observed with certainty in a single run is very limited. The algorithm relies strongly on the ability of MRF beliefs to generalize observations. The POMDP planner evaluates the benefit of each exploration move by measuring how the possible outcomes can be usefully generalized.

Preprocessing beliefs: Computing the marginal distributions $\Pr(\omega_x \mid b)$ of belief functions is an important step of the algorithm. The marginal distributions of b are necessary for two purposes: (i) determining the safe flight envelope model $\bar{\rho}(b)$ used to compute ordinary actions success probabilities; (ii) determining macro-actions transition probabilities from b, that is, set of beliefs that can be reached from b through a macro-action, and their probabilities. As explained before, computing marginals is intractable and we use the MCMC technique known as Gibbs sampling to ap-

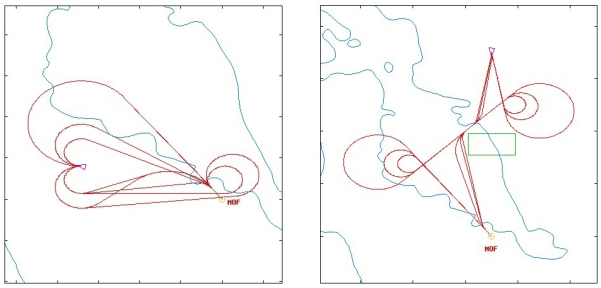

Figure 2: Depending on the outcome of exploration, the plane (triangle) follows a different route to Moffet Field (MOF). Left: The plane first explores right bank angles and positive descent rates. If this exploration is successful enough, it lands following the shortest path. Otherwise, it searches for an alternative route by exploring left bank. Right: The planner finds different way to go around a rectangle obstacle. If there is insufficient right (resp. left) bank, one of the large loops on the right (resp. left) of the figure is executed.

proximate them. Then we try to limit the number of beliefs reachable through the use of a finite set of macro-actions, relying on MRFs generalization abilities to compensate. Nevertheless, our preliminary simulation results show that the vast majority of the algorithm execution time may be spent in Monte Carlo simulations, and that it sometimes blows up the computation time above the delays admissible for emergency landing planning (which is in the order of 10 seconds). This cost can be reduced by performing less simulation trials, which reduces the accuracy of the estimates. More interestingly, we observe that the computation of marginal probabilities is independent of the plane location. Therefore, if the initial belief is known in advance, then all probabilistic inferences involving beliefs can be performed off-line, before knowing the initial situation from which the emergency landing planner will be run. More precisely, we can compute in advance the set of reachable beliefs, the macro-actions belief transition probabilities, and the safe flight envelope models $\bar{\rho}(b)$ for each reachable belief b. This information is saved to a file and all other belief related data can be discarded (including all marginal probability estimates). Later, this information can be quickly reloaded and used to develop the AO* search tree from a given initial situation. In this way, the on-line computation time the algorithm is reduced to fraction of the total execution time.

Implementation

We have implemented a preliminary version of our POMDP planner and connected it to the real-size emergency landing planner developed in the IRAC project. It was made possible through the following model simplifications:

- The air speed variable is removed from the robot configurations used for path planning. It is not a control anymore. Instead, it is decreased uniformly over each trajectory;
- The flight envelope is represented by a two-dimensional MRF defined over the bank angle (which is closely related to turn rate) and descent rate. The value of a control state is supposed independent of the altitude and speed;
- Simplifying assumptions are used to compute macro-actions transition probabilities.

Our preliminary implementation uses the simple heuristic h_{Euclid} instead of h_{optimist}. Figure 2 shows examples of contingent plans. The most complex plan (right) was computed in about 17s on an average laptop computer, around 19% of it being devoted to belief computation. The cost of building the probabilisitc roadmap appears neglectible, so, the reamining 81% are spent mostly in AO* search. This is because collision checking is performed in a lazy way, at the time of searching de graph (an edge is computed only if is traversed by search). Indeed, 72% of AO* search time is spent in collision checking. Execution time blows if we increase the number of Monte Carlo samples and the number of waypoints in the roadmap. Future work will evaluate how various acceleration techniques discussed above help address this complexity.

Conclusion

We outlined a new class of problem involving motion planning with uncertain non-holonomic constraints. Through the example of damaged aircraft, we showed an approach based on carefully selecting the state variables that are represented at each stage of reasoning (exploration and path planning), and ensuring a coherent interaction of different models (MRF and POMDP). Various optimization techniques, such as powerful heuristics, intelligent macro-actions, and belief preprocessing allow applying this approach to real-size problems. Future work will present the technical details hidden here, as well as extensive simulation results.

Acknowledgments

This work was supported by the Intelligent Resilient Aircraft Control program of the NASA Aeronautics Research Mission Directorate. We thank Ella Atkins and Robert Sanner for discussion on their work, and Christian Neukom for feedback on this paper.

References

Casella, G., and George, E. I. 1992. Explaining the gibbs sampler. *The American Statistician* 46(3):167–174.

Choset, H.; Lynch, K.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L.; and Thrun, S. 2004. *Principles of Robotic Motion: Theory, Algorithms, and Implementation*. MIT Press.

Hauskrecht, M.; Meuleau, N.; Kaelbling, L. P.; Dean, T.; and Boutilier, C. 1998. Hierarchical solution of Markov decision processes using macro-actions. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 220–229.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1-2):99–134.

Kindermann, R., and Snell, J. L. 1980. *Markov Random Fields and Their Applications*. American Mathematical Society.

Meuleau, N.; Plaunt, C.; Smith, D.; and Smith, T. 2009. A comparison of risk sensitive path planning methods for aircraft emergency landing. In *ICAPS-09: Proceedings of the Workshop on Bridging The Gap Between Task And Motion Planning*, 71–80.

Mitchell, T. M. 1997. *Machine Learning*. McGraw-Hill.

Tang, Y.; Atkins, E.; and Sanner, R. 2007. Emergency flight planning for a generalized transport aircraft with left wing damage. In *Proc. Guidance, Navigation, and Control Conference*. AIAA.

Thrun, S.; Burgard, W.; and Fox, D. 2005. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. MIT Press.

Yi, G., and Atkins, E. 2010. Trim state discovery for an adaptive flight planner. In *Aerospace Sciences Meeting*. AIAA.