Observations on SOFIA Observation Scheduling: Search and Inference in the Face of Discrete and Continuous Constraints

Jeremy Frank and **Michael Gross**[*] and **Elif Kürklu**[†]
NASA Ames Research Center
MS N269-3
Moffett Field CA 94035-1000
{frank,ekurklu}@email.arc.nasa.gov, mgross@mail.arc.nasa.gov

Abstract

We did cool stuff to reduce the number of IVPs and BVPs needed to schedule SOFIA by restricting the problem. The restriction costs us little in terms of the value of the flight plans we can build. The restriction allowed us to reformulate part of the search problem as a zero-finding problem. The result is a simplified planning model and significant savings in computation time.

Introduction

The Stratospheric Observatory for Infrared Astronomy (SOFIA) is NASA's next generation airborne astronomical observatory. The facility consists of a 747-SP modified to accommodate a 2.5 meter telescope. SOFIA is expected to fly an average of 140 science flights/year over it's 20 year life time. The SOFIA telescope is mounted aft of the wings on the port side of the aircraft and is articulated through a range of 20 to 60 degrees of elevation. The telescope has no lateral flexibility; thus, the aircraft must turn constantly to maintain the telescope's focus on an object during observations. A significant problem in future SOFIA operations is that of scheduling Facility Instrument (FI) flights in support of the SOFIA General Investigator (GI) program. GIs are expected to propose small numbers of observations, and many observations must be grouped together to make up single flights. Approximately 70 GI flight per year are expected, with 5-15 observations per flight. The scope of the flight planning problem for supporting GI observations with the anticipated flight rate for SOFIA makes the manual approach for flight planning daunting.

Automated flight planning for SOFIA involves selecting observations to perform and scheduling these observations. These discrete choices are constrained by complex continuous constraints. Verifying that the constraints are satisfied involves solving both Initial Value problems (IVPs) and Boundary Value Problems (BVPs) to find the aircraft's ground track, determine aircraft fuel consumption, and check that observations stay within proscribed elevation limits. A sampling based approach called ForwardPlanner (FK03) works well for a simple SOFIA model, but is too costly for a higher fidelity model. The main reason for this is the high cost of solving many IVPs and BVPs in the higher fidelity model. A set of well-founded assumptions allowed us to reformulate the model and eliminate a large number of calls to solve the IVPs and BVPs; while this reformulation actually reduces the search space, empirical results show that the resulting algorithm produces high quality flight plans at a fraction of the computational effort.

The rest of the paper is organized as follows. We first describe the high fidelity SOFIA model. We then re-examine the ForwardPlanner algorithm and describe a principal source of the increased computational costs of flight planning. We then describe a way of migrating some of the search into the underlying constraint reasoning component by means of some well-founded assumptions. This allows us to eliminate a potentially large number of expensive ground track construction steps without sacrificing the ability to construct good flight plans. We perform several experiments to validate the approach. Finally, we discuss the implications of our reformulation of the computational search on the planning model.

Improving Model Fidelity

The SFPP (Single Flight Planning Problem) consists of a number of observation requests, a flight day, and a takeoff and landing airport. The objective is to find a flight plan that maximizes the summed priority of the observations performed while obeying the constraints governing legal flights. We also presented ForwardPlanner, a progression style planner using a combination of lookahead and heuristics to solve the resulting SFPP. The aircraft activities are *take-off, land, flight-leg* and *dead-leg*. Flight-legs require tracking an object and obeying visibility constraints, while dead-legs can be used to reposition the aircraft to enable flight-legs, and only consume time and fuel.

Insert formal description of math here. Trying to do this with my present formulation is awful! What I'd really like is the heading as a function of time (with initial position and object coords as constants), same for elevation. Uncertain if cartesian formulation better or worse?

[*]University Space Research Association
[†]QSS Group, Inc.

We summarize the relevant constraints on the problem, originally described in (FK03). As before, we use the astronomical conventions established in Meuss (Mee91). The constraints on object elevation are as follows. Suppose that ϕ_i is the initial latitude, L_i is the initial longitude, θ_i is the (Sidereal) time at which the leg begins, θ_f is the time at which the leg terminates, δ is the observation declination, α is the observation Right Ascension, b_i is leg heading at the start of the leg, b_f is the heading at the end of the leg, h is the object elevation (as a function of object coordinates, position and time), A_f is the object azimuth at the end of the leg (again, a function of object coordinates, position and time), and d is the flight distance. Assume that the Earth is a perfect sphere, and that the aircraft flies a Great Circle segment. The end position of the aircraft after the leg is given by: **check these equations, reverse engineered from sphtr**

$$\phi_f = \sin^{-1}\left[\sin(\phi_i)\sin(\frac{\pi}{2}-d)+\cos(\phi_i)\cos(\frac{\pi}{2}-d)\cos(b_i)\right] \tag{1}$$

$$L_f = L_i + \tan^{-1}\left[\frac{\sin(b_i)\cos(\frac{\pi}{2}-d)\cos(\phi_i)}{\sin(\frac{\pi}{2}-d)-\sin(\phi_i)\sin(\phi_f)}\right] \tag{2}$$

The object elevation is given by:

$$H \equiv \theta_f - L_f - \alpha \tag{3}$$

$$\sin h = \sin(\phi_f)\sin(\delta)+\cos(\phi_f)\cos(\delta)\cos(H) \tag{4}$$

The object azimuth is given by:

$$\tan A_f = \frac{\sin(H)}{\cos(H)\sin\phi_f - \tan(\delta)\cos(\theta_f)} \tag{5}$$

Finally, the aircraft heading is given by:

$$b_f = \tan^{-1}\left[\frac{\sin(b_i)\cos(\phi_i)\cos(\frac{\pi}{2}-d)}{\sin(\phi_i)-\sin(\frac{\pi}{2}-d)\sin(\phi_f)}\right] \tag{6}$$

Here's the nonlinear DEs form of this; Cartesian formulation better, but above equations more intuitive for defining the zero-finding problems.

Since the telescope has no lateral flexibility, the aircraft's ground track is constrained by the requirement to track the object. The telescope points out the left hand side of the aircraft, so the constraint is $b_f + 270° = A_f$. In addition, the telescope elevation h is limited to between 20° and 60°. Both of these conditions must hold throughout the observation. The former actually constrains the ground track, leading to a system of differential equations defining the ground track. The latter constraint is a condition on the elevation that must be satisfied. We write the equations of motion for the aircraft assuming a spherical Earth of radius a and constant ground speed V. If we interpret A and h as functions of (ultimately) time, we can then write a system of differential equations as follows:

$$\frac{d}{d\theta}\phi(\theta) = -\frac{V\cos(A(\theta)-r)}{a} \tag{7}$$

$$\frac{d}{d\theta}L(\theta) = -\frac{V\sin(A(\theta)-r)}{a\cos\phi(\theta)} \tag{8}$$

By virtue of simple physical intuition, there is a unique solution to these equations; that is, there is only one ground track the aircraft could follow in order to track an object of known and fixed α and δ beginning at a known initial position and initial time. We can rewrite the elevation equations as a function of time:

$$\sin h(\theta) = \sin(\phi(\theta))\sin(\delta)+\cos(\phi(\theta))\cos(\delta)\cos(\theta-L(\theta)-\alpha) \tag{9}$$

Similarly, we know from physical intuition that the change of elevation described by this function is continuous and differentiable with respect to time. That is, the elevation equation is constrained by the function governing the aircraft's position, which we similarly argued was well behaved. Thus, we can compute $\frac{dh}{d\theta}$ and, armed with the functions $\phi(\theta)$ and $L(\theta)$, find the solutions to $\frac{dh}{d\theta}=0$.

The set of constraints considered in the previous paper comprise a simplified version of the problem. In particular, the following factors were ignored:

- The impact of the true fuel consumption model of the aircraft on the flight time. Previously, we simply used a maximum flight duration as an analog of fuel consumption. The fuel consumption is actually a function of aircraft weight, Mach number, change in altitude, and outside temperature. Since flying reduces aircraft weight, the flight duration constraint in the old model is replaced by a partial differential equation that governs the fuel consumption.
- The impact of the Earth's shape on the ground track. The Earth is actually an oblate spheroid whose polar diameter and equatorial diameter are not the same. This has a reasonable impact on the actual ground track, and accounting for this invalidates the differential equations in the previous model that constrain the ground track.
- The impact of winds on the ground track. As the aircraft flies, the wind direction and velocity changes the ground speed. This invalidates the assumption in the old model that the ground speed is constant.

In addition, we previously Euler's method (Fer81) to solve the (simplified) aircraft dynamics equations; while fast when assuming a spherical Earth, it is prone to error, and harder to use without the spherical Earth approximation and deviation due to winds. These constraints are solved using 5^{th}-order Runge-Kutta approximation (Fer81) which calculates ground track segments on the surface of the Earth; this approach constructs only as many Great Circle segments as necessary to keep the error within a fixed tolerance. **Mike, this is probably wrong: help fix this.** A gridded wind and temperature model is available to correct the ground track in the face of winds and provide temperature data for calculating fuel consumption. In addition, an aircraft performance model from Boeing is used to compute the exact fuel consumption for each of the 747-SP's engines, providing a much better estimate of flight time and correct fuel consumption. Additional features include the ability to track non-stellar objects such as the sun, moon, planets in the solar system, and comets, as well as corrections for the required aircraft heading due to aircraft pitch.

Unfortunately, the costs of solving these new constraints and the higher accuracy of the flight dynamics constraints lead to a serious degradation in computational efficiency. Using the new reasoning system, the ForwardPlanner algorithm takes roughly 300 times as long to build a flight plan than it does using the simpler constraints and constraint reasoning system.

Explaining The Performance Hit

Our investigation into the ForwardPlanner algorithm revealed that we spend a considerable amount of time deciding which observations are *feasible* at any point in the planning process before evaluating them. An observation o is feasible at time t and position p if there is a *dead-leg* of possibly zero duration that ensures that the observation is within the elevation limits at t', p', the observation stays within the elevation limits for the required duration of the observation, and the aircraft can fly to the landing airport after the observation is finished. If the observation is not visible at t, p, ForwardPlanner performs a search for the *shortest dead-leg* that makes the object visible for a flight leg of the required duration, and still allows the aircraft to fly home after the observation is finished. This search is done by first changing the latitude of the aircraft to make the object visible, then performing a brute force search to reduce the dead-leg duration. If the resulting dead-leg exceeds a bound D the observation is considered infeasible. Each flight-leg and dead-leg construction step requires solving an IVP, while each check to ensure the aircraft can flight to the landing airport requires solving a BVP.

get data on how many dead-leg search steps are usually performed In the worst case, this requires ForwardPlanner to solve a very large number of IVPs and BVPs. This is true even though the dead-leg duration is limited, as are the heading choices for the enabling dead-leg. It is also worth noting that the discretization of dead-leg headings and durations in ForwardPlanner is primarily for computational efficiency, and can result in missing some dead-legs that enable observations. With the increased computational expense of each leg construction step, it is important to reduce the number of leg construction steps as much as possible. At the same time, we would like to eliminate the discretization of dead-leg parameters. We focus on this throughout the rest of the paper.

Before moving on, there are a number of points worth making. First, the shortest dead-leg making the object visible immediately after the dead-leg may not make the object visible for long enough. Suppose the aircraft is at high latitude. It is possible to fly West towards an object that is setting and make this object appear to rise. Observing the object will require flying perpendicular to the object, thus making it appear to set again. It is easy to construct a case where the aircraft may need to fly a longer dead-leg to enable an observation of the right duration. Such an object would have to be valuable to justify adding it to the flight plan; however, recent studies indicate that such Northerly flights are likely to be common, so this is a case worth bearing in mind. Similarly, the shortest dead-leg making the object visible for long enough may not enable the aircraft to fly home after the observation is completed. However, this only happens if the flight is almost finished. Thus, failing to establish this condition may lead to missing only one observation; the likelihood that this observation is critical to making the flight a good one is low, and is not as important a consideration as the previous issue.

Handling the Higher Fidelity Model

In this section we describe how to change the solution methodology to reduce the cost of finding plans without sacrificing performance. First we describe a modification to the ForwardPlanner that *restricts* the set of plans that can be built, and show empirically that this leads to an increase in speed without sacrificing performance. We then show how to leverage this change to get an even larger increase in speed, again without sacrificing performance.

Restricting the Set of Plans

The feasibility check may requires a large number of expensive BVP checks to ensure that the aircraft can return to the landing airport. If the shortest dead-leg enabling an observation makes it impossible to return home, we consider *lengthening* the dead leg in the hopes that the resulting flight leg will carry the aircraft *closer* to the landing airport. This is not as counter intuitive as it seems, due to the complexities of the ground track. However, it may be a waste of time, since the aircraft will trivially be in range of the airport for at least half the flight [1]

We can *restrict* the feasibility check in the following way: first, we find the shortest dead leg that enables the observation for the desired duration. If the aircraft can return to the landing airport after completing both this dead-leg and the observation, then the observation is feasible, otherwise it is not feasible. It might be possible to find a longer dead-leg that allows the aircraft to return to the landing airport, thus using this policy will exclude some flight plans. However, these will generally be confined to the latter half of the flight; furthermore, we expect a performance degradation only when *high priority* observations are excluded.

figure showing this works goes here

[1]SOFIA's nominal operation has the aircraft take off and land at the same airport; this assumption only holds in such cases.

beef this up a little Once we have committed to this policy, we can introduce another performance enhancement. ForwardPlanner requires collecting all of the feasible observations, heuristically ranking them, and subsequently selecting one to add the flight plan. This requires solving all of the BVPs up-front. However, we can *postpone* the solution of the BVPs until *after* deciding to add an observation to the flight plan. If the observation chosen to extend a flight plan fails to satisfy the requirement, then it is discarded and another extension is chosen. This requires caching the enabling dead-leg with the observations in order to avoid searching for them again, and also requires re-normalizing the distribution and re-sampling. However, the probability of choosing any feasible extension of the flight plan is unchanged, and the expected number of BVPs to solve is reduced when most of the extensions are feasible **and annecdotally this is the case; validate this with Elif.**

figure showing this works goes here?

Changing the Division of Labor

We have shown how to reduce the number of BVPs that must be solved to produce good flight plans. However, even with this improvement, brute force search is still required to find the shortest dead-leg that enables the observation. In this section, we show how to eliminate the brute-force search. We take advantage of the new restricted feasibility condition by defining a function whose zeros correspond to the properties of an approximation to the shortest dead-leg enabling the observation. This defines a sub-problem that can be efficiently solved by using zero-finding algorithms such as Newton's Method. Because the resulting formulation allows us to search the full continuous space of dead-legs, this is an improvement over the discretized brute-force search done in the previous version of ForwardPlanner.

Replacing Search With Zero Finding

Using the restricted conditions on object feasibility, the dead-leg construction phase of the feasibility check requires finding the heading and duration of the shortest dead-leg that enables the observation for a sufficient amount of time. Most often, this occurs because an object is not initially within the elevation limits. However, sometimes it occurs because an object violates the elevation limits during the flight-leg, even when it is initially visible.

figure needed? Let us consider the *feasible region* of an observation o. This region is the set of positions on the Earth from which the observation is visible, and is the annulus defined by two circles centered at the nadir position of o whose radii are the coelevation limits of the telescope (in SOFIA's case, the radii of these circles are 30 and 70 degrees of arc). Let us now consider the properties of the shortest dead-leg in terms of the feasible region. Initially the aircraft is outside the feasible region. We want the aircraft to be in the feasible region after completing the dead-leg. Now, the *shortest* leg would put the aircraft on the boundary of the feasible region, as opposed to anywhere strictly inside it. This corresponds to a condition that the object elevation being at one of the two extremes. If the aircraft begins inside the inner circle of the annulus, then we want the object to be precisely at the the upper telescope elevation limit, while if it is outside the outer circle, we want the object to be at the lower telescope elevation limit.

If the object was fixed relative to the ground, we could simply fly directly towards the object, since this maximizes the rate of change of the object elevation. However, as we mentioned, the object appears to move across the Earth as the Earth rotates. We could fly a dead-leg that tracks the object as it moves, but that would not minimize the flight distance. We know that a Great Circle arc minimizes the flight distance, but what Great Circle arc should we follow? We use the following intuition: we fly a Great Circle arc that *ends* with the aircraft flying directly towards the object to be observed. Intuitively, this is the correct policy when the object is nearly in view, or near the end of longer dead-legs. Observatory policy will normally prevent dead-legs longer than a few tens of minutes, so this intuition will likely produce very short, if not "locally optimal" dead-legs.

Thus, we have the following problem: find b_i, d such that $F_1(b_i, d) =< f_1(b_i, d), f_2(b_i, d) >=< 0, 0 >$ where $f_1(b_i, d) = b_f - A_f$ is the difference between the object azimuth and the final heading of the aircraft after flying the dead-leg defined by b_i, d, and $f_2(b_i, d) = e - h$ is the difference between the final object elevation and the telescope elevation limit e closest to the initial object elevation. Equations 1 to 6 show how to compute all of the quantities needed to define F_1.

Now let us consider the case where the object violates the elevation limits at some point during the observation, regardless of whether or not it is initially visible. Using the geometric interpretation of the feasible region again, we see that the flight track exits the annulus (and possible re-enters it later on). In this case, we can set up a function very similar to that we used when the observation was initially outside the feasible region. We now want to find b_i, d such that $F_2(b_i, d) =< f_1(b_i, d), f_3(b_i, d) >=< 0, 0 >$, where $f_3(h, d)$ is the difference between the *extreme* object elevation achieved during the flight-leg and the telescope elevation limit violated during the observation. The intuition behind this is that the dead-leg we wish to fly should just barely nudge the observation inside the feasible region. f_1 remains the same. Unlike the previous case, where we only needed to compute quantities like position and object elevation at fixed times, we now must find either the minimum or maximum of the elevation over the course of the flight-leg. If the extreme elevation occurs at either the beginning or end of the flight leg, this only requires evaluating Equation 4. Otherwise, it requires a more expensive function optimization step **Mike, what algorithm used to do this? how expensive in practice?**

In both cases, we have now reduced the problem of finding the shortest dead-leg to the problem of finding a zero of a function, which can be solved efficiently using a variety of methods as long as F_1 satisfies some simple conditions ??. If an object is not initially visible, we determine whether to zero F_1 or F_2 using **Elif will fill in details**. In some cases, we simply don't bother looking for a dead-leg since we are certain not to find one; these are cases where the extreme elevation is out of bounds and the rate of change of the elevation changes **provide better description of this case, and explain why we don't bother. Perhaps move this to another section?**

Properties of the dead-legs Let us now consider the zeros of the functions F_1 and F_2 and the dead-legs that are defined by them. We want to do this for two reasons. First, the behavior of zero-finding algorithms depends on how many zeros there are and how they are distributed. Second, the resulting dead-legs may not be feasible given other constraints on how the aircraft flies that are not present in the definition of the zero-finding problems.

First of all, we observe that there are an infinite number of zeros of both F_1 and F_2. This is because we have imposed no restriction on b_i and d. In particular, we can always find a sequence of increasingly long dead-legs that bring the object to an extreme elevation limit. This does not pose a serious problem, because of the dead-leg duration restriction imposed by the Forward-Planner algorithm. However, it means that we might not find the shortest dead leg, and thereby incorrectly conclude that some observation is not feasible.

Also, not all zeros correspond to valid dead-legs. For example, a dead leg whose duration is negative is impossible for the aircraft to fly; similarly, a dead-leg whose duration exceeds the maximum allowed is forbidden. A more interesting example concerns limitations imposed by the minimum turn duration of the aircraft. A standard rate turn for a 747 is 180 degrees in 2 minutes. If the heading change and duration of the dead-leg violate this constraint, then the minimum dead-leg is also impossible to achieve.

Despite these drawbacks, we should point out that this method has two significant advantages over the brute force approach we used previously. First, we have imposed no limitations on the heading or durations of the dead-legs. Thus, we might find dead-legs we were unable to find before using this new method. Second, since zero-finding algorithms are usually quite fast, we hope that employing such a method will dramatically speed up the feasibility check, and therefore the flight planning algorithm overall.

Finding dead-legs By Zeroing

Newton's Method and Ratio of Determinants Update

Newton's Method is our choice for finding the zeros of F_1 and F_2. It is simple to implement and very fast. Newton's Method requires an initial guess for the zero; let this be denoted b_1, d_1 with future iterates denoted b_i, d_i . For functions F of 2 inputs and 2 outputs, the method proceeds as follows:

1. Compute $F(b_i, d_i) =< f_1(b_i, d_i), f_2(b_i, d_i) >=< f_1, f_2 >$
2. Compute the Jacobian (matrix of partial derivatives):
$$J = \begin{pmatrix} \frac{\partial f_1}{\partial b}(b_i, d_i) & \frac{\partial f_1}{\partial d}(b_i, d_i) \\ \frac{\partial f_2}{\partial b}(b_i, d_i) & \frac{\partial f_2}{\partial d}(b_i, d_i) \end{pmatrix} = \begin{pmatrix} p & q \\ r & s \end{pmatrix}$$
3. Compute the determinant of J : $|J| = ps - qr$. If this is smaller than t then set $|J| = t$ (preserving the sign).
4. Compute the ratio of determinants update: $db = \frac{f_2 q - f_1 s}{|J|}$ and $dd = \frac{f_1 p - f_2 r}{|J|}$
5. Set $b_{i+1} = b_i + db$ and $d_{i+1} = d_i + dd$
6. If $< b_{i+1}, d_{i+1} > \approx < 0, 0 >$ or step limit reached, then halt, otherwise go to step 1.

Computing Derivatives Numerically

Directly calculating the derivatives of the functions F_1 and F_2 is impossible because of the gridded wind model that influences the ground track. Consequently, we use finite differencing to compute all of our derivatives numerically (GMW81). Of the available schemes, we chose forward differencing over centered differencing because of the smaller number of function evaluations required. We use two step size parameters s_1, s_2 in forward differencing. Suppose we are computing the derivative at b_i, d_i. Forward differencing for functions F of 2 inputs and 2 outputs, the method proceeds as follows:

1. Compute $F(b_i, d_i) =< f_1(b_i, d_i), f_2(b_i, d_i) >=< f_1, f_2 >$
2. Compute $f_{1b} = f_1(b_i + s_1, d_i)$
3. Compute $f_{1d} = f_1(b_i, d_i + s_2)$
4. Compute $f_{2b} = f_2(b_i + s_1, d_i)$
5. Compute $f_{2d} = f_2(b_i, d_i + s_2)$
6. Compute $\frac{\partial f_1}{\partial b} = \frac{f_{1b} - f_1}{s_1}$
7. Compute $\frac{\partial f_1}{\partial d} = \frac{f_{1d} - f_1}{s_2}$
8. Compute $\frac{\partial f_2}{\partial b} = \frac{f_{2b} - f_2}{s_1}$
9. Compute $\frac{\partial f_2}{\partial d} = \frac{f_{2d} - f_2}{s_2}$

Note that more elaborate forms of numerical derivative computations are available. One reason for avoiding them is the number of calls to compute F_1 or F_2, which in this case requires constructing either flight legs, dead-legs or both. Since we want to minimize this cost, for the time being we stick with the simple forward differencing scheme.

Zeroing F_1 only requires solving 1 IVP to actually construct the flight-leg. Zeroing F_2 requires solving two IVPs per step of Newton's Method, and two function optimization steps to find the elevation extremes, plus one more at the end to construct the flight. Thus, we depend on performing only a small number of Newton steps to increase the speed.

The Initial Guess

Still need some clarification on this section Algorithms like Newton's Method are highly sensitive to the closeness of the initial guess to the actual zero of the function. Since these algorithms typically follow gradients towards local zeros, it is important to ensure that the initial guess is a good one.

Guessing the initial dead-leg duration requires estimating the difference in elevation that the dead-leg must achieve, and then estimating the rate of change of the elevation during the dead-leg. Guessing the initial heading requires determining how an object's elevation is changing, and choosing the flight direction to make the elevation change correctly.

This can't be as complex as it appears...Mike, Elif? Suppose we compute the ground track for a flight leg. We can then determine the extreme of the elevation by estimating the form of Equation 9 and decide whether to zero F_1 or F_2. We can also use Equation 9 to approximate the rate of change of the elevation as a function of time; this is done by taking the derivative of Equation 9 and evaluating it at the position and time at which the flight leg begins. We can also estimate the rate of change of the elevation as a function of the change in position, which requires estimating the derivatives of Equations 1 and 2 as a function of time at the current position and time as well.

We attempted to improve convergence when zeroing F_2 by first using Euler's Method to construct the flight legs. Once a zero of F_2 was found this way, we then used this as an initial guess and re-ran Newton's Method using Runge-Kutta to construct the flight legs. This did not improve convergence and was more expensive, and so we do not consider this further.

Matters of Convergence

Newton's Method depends on the function being zeroed to obey some properties to guarantee convergence. Our functions do not obey these properties all of the time, and so Newton's Method occasionally fails to converge.

First, there may be cases where the first derivative of F_1 or F_2 may be zero. This is problematic for zero-finding algorithms, since they mostly rely on the first derivative to provide the direction of the next move. **Ask Mike if we have concrete evidence that this happens.**

Ask Mike for clarifying details on compact support for newton. Newton's Method requires that F is defined on every element of $\mathcal{R}^2$. As we have seen above, this is not the case. F is not well defined for sufficiently short or long dead-leg durations. The problem with long durations is due to the built-in nature of the fuel model, since fuel consumption and aircraft weight are so intertwined. Essentially, if a Newton step requires the aircraft to fly long enough that it would run out of fuel, we can't evaluate the ground track of the flight-leg [2]. The problem with short durations has been explained above.

Finally, since we have only discrete approximations of the continuous functions that define the wind speed and direction, we know that F_1 and F_2 are not actually continuous and differentiable functions as implemented inside the constraint reasoning system. Thus, it is possible that the zeros will be badly behaved because of this approximation as well.

These factors mean that convergence of Newton's method may be interrupted if any intermediate step violates one of these conditions. This is a problem because it is conceivable that the zero found by Newton's method can correspond to a legitimate supporting dead-leg even if an iteration of Newton's method corresponds to a senseless dead-leg. If the function or the derivatives can't be evaluated during Newton's Method, our only option is to truncate the feasibility check and report that the observation is not feasible. Additionally, we could find Newton's Method failing to converge or converging after a large number of steps; we thus use a cutoff value to terminate search.

Empirical Results

We evaluated the ideas outlined above using several experiments.

1. relaxing the feasibility check does not cost in terms of goodness of plans
2. number of times newton's method fails to find a good dead-leg when brute did
3. number of newton's violating a condition during iteration when a sensible dead leg could result (difficult to test all of these because of model changes)
4. speedups due to Newton's method for relaxed feasibility check
5. impact of variations on Newton (cutoff for too long or too short/negative steps, error, etc.)

Impact on Planning Model

In this section we discuss the impact the above changes have on the underlying model used by the planner. Previously, the planner had to perform an explicit search for dead-legs for observations. However, by using the above tools, we can now write a deterministic procedure that maps a permutation of observations and a takeoff time into a legal flight plan and a set of observations that were not performed. This simplifies the planning model a great deal, and makes it easier to consider algorithms like GAs, SA and SWO that operate directly on the permutation space. As we have said above, the continuous model is not perfect, but is good enough.

It is possible to make too much of this. For instance, the brute force search of the original ForwardPlanner would find the optimal shortest dead-leg if it could search over the full range of headings. Thus, choosing to view this component as a part of the planner

[2]Curiously enough, the problem with negative durations does not exist with the simple flight dynamics constraints; these are solved using spherical triangles, and a negative flight duration simply changes one of the coordinates of the triangles. Thus, we could try replacing failed evaluations of f_3 with an Euler's method approximation, but we didn't.

instead of an incomplete procedural constraint is something of a matter of taste. It is also straightforward to build GA, SA and SWO based algorithms on the same components as the ForwardPlanner is built on. However, the machinery described in this paper makes these options much more palatable (at least to me.)

Complete search is now actually feasible if the number of start times is not too high. Annecdotal evidence indicates that this may be true.

Conclusions and Future Work

In this paper we *restricted* the feasibility check to reduce the number of IVPs and BVPs to solve. We can also *relax* the definition of feasibility and drop the check on the return to the landing airport altogether. The consequence of doing this is that a flight plan may violate the fuel constraint and either need to be repaired or rejected completely. This may be justified because throughout most of the planning process this condition is trivially satisfied. However, the costs of repair and rejection sampling were deemed too high at this time. Relaxing the feasibility check by only ensuring that the observation is visible immediately after a dead-leg is a more dangerous proposition, because the frequency of high latitude observing almost ensures rejection sampling or repair will be needed.

We should point out that we can define an f_4 which accounts for the difference between the amount of fuel remaining to get the aircraft to the landing airport and the amount of fuel consumed by the leg home. Doing so would allow us to use the more restrictive definition of feasibility while paying only minimal overhead in the number of BVPs and IVPs that we must solve. However, no method of doing so we have yet discovered avoids the pitfalls described previously. Letting $f_4 = 0$ if the landing airport is reachable creates many situations where the first derivative of f_4 is zero, which is bad. Forcing $f_4 = 0$ only if exactly enough fuel is left to get to the landing airport is equally bad.

The condition on dead-legs is "locally optimal" in the sense that it is the best action to support one observation. However, it may adversely affect the rest of the plan. Currently, we rely on repeated sampling to find good plans, but we know we only sample some of the possible plans,and may miss the best possible plan, Extensions of the functions to zero may fix this but it's hard to see what to do.

We intuitively define criteria to minimize the dead-leg duration (e.g. dead-leg ends with us flying towards the object). We haven't proved this is shortest even on spherical earth with no winds, but believe it's correct. Other criteria might be better given that we must account for winds and the like.

make sure to reference boddy and johnson or boddy and krebsbach as appropriate.

References

J. Ferziger. *Numerical Methods for Engineering Applications.* John Wiley and Sons, 1981.

J. Frank and A. JónssonE. Kürklu. Sofia's choice: Scheduling observations for an airborne observatory. *Proceedings of the* 13th *International Conference on Automated Planning and Scheduling*, 2003.

P. Gill, W. Murray, and M. Wright. *Practical Optimization.* Academic Press, 1981.

J. Meeus. *Astronomical Algorithms.* Willmann-Bell, Inc., 1991.