Applying Squeaky-Wheel Optimization Schedule Airborne Astronomy Observations

Jeremy Frank and **Elif Kürklü***
NASA Ames Research Center
Mail Stop N269-3
Moffett Field CA 94035-1000
{frank,ekurklu}@email.arc.nasa.gov

Abstract

We apply the Squeaky Wheel Optimization (SWO) algorithm to the problem of scheduling astronomy observations for the Stratospheric Observatory for Infrared Astronomy, an airborne observatory. The problem contains complex constraints relating the feasibility of an astronomical observation to the position and time at which the observation begins, telescope elevation limits, special use airspace, and available fuel. Solving the problem requires making discrete choices (e.g. selection and sequencing of observations) and continuous ones (e.g. takeoff time and setting up observations by repositioning the aircraft). The problem also includes optimization criteria such as maximizing observing time while simultaneously minimizing total flight time. Previous approaches to the problem fail to scale when accounting for all constraints. We describe how to customize SWO to solve this problem, and show that it finds better flight plans, often with less computation time, than previous approaches.

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 per year over it's 20 year lifetime, and will commence operations in early 2005. 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° of elevation. The telescope has minimal 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, called the SFPP (Single Flight Planning Problem). 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. There has been considerable success in automating observation scheduling for ground-based telescopes (Bresina 1996), space-based telescopes such as Hubble Space Telescope (Johnston & Miller 1994), Earth Observing Satellites (Potter & Gasch 1998) and planetary rovers (Smith 2004). However, the SOFIA flight planning problem differs from these problems in a variety of ways. Observations are feasible over large, continuous regions of space and time. The motion of the aircraft is governed by differential equations, and the aircraft can be flown in any direction for any length of time to enable an observation. The principal feasibility condition for observations is a nonlinear function over the solution to the equations of motion. As a consequence of these factors, even though SOFIA has a "closed tour" constraint that makes it appear similar to problems such as the Traveling Salesperson Problem, there are no fixed waypoints to define routes. Also, the SOFIA problem cannot be characterized only by discrete decisions. The complexity of the differential equations and feasibility constraints makes it difficult to find good heuristics, and the expense of solving the differential equations impacts solver performance.

Previously, a combination of heuristic search, biased stochastic sampling, approximations and continuous optimization methods were used to produce an algorithm called ForwardPlanner for the SFPP (Frank & Kürklü 2003; Frank, Gross, & Kürklü 2004). However, this approach ultimately fails to scale as more and more constraints on valid flight plans are added to the problem description. Computationally expensive lookahead search is needed to obtain good results from ForwardPlanner. While approximations reduce the costs of lookahead, they inaccurately calculate altitude-sensitive fuel consumption, ignore the impact of winds, and fail to account for special-use airspace limitations, leading to poor quality flight plans. Consequently, a new approach to the problem is needed.

The rest of the paper is organized as follows. We first formally describe the SFPP, the constraints on flight plans, and the optimization criteria used to compare valid flight plans. We then briefly describe the ForwardPlanner and discuss its problems. We then introduce Squeaky Wheel Optimization (SWO) and discuss how to apply it to the SFPP. We show that SWO improves upon ForwardPlanner on a small set of examples. We then discuss a variety of ways to improve the performance of SWO. We describe experiments to validate the approach. Finally, we conclude and discuss future work.

*QSS Group, Inc.

SOFIA's Choice

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. The aircraft can perform two different classes of activities during a flight. *Flight-legs* require tracking an object and are only legal if the object is within the telescope elevation limits throughout the observation. *Dead-legs* can be used to reposition the aircraft to enable flight-legs, but no observations are performed. A distinguished class of dead-legs are used to take off and return to the landing airport.

The input to the SFPP consists of a set of observation requests, each consisting of the Right Ascension (RA) α and Declination (Dec) δ, observation duration, priority; a flight date; initial fuel load; an altitude profile; earliest takeoff time θ_l and latest landing times θ_u; the designated takeoff and landing airports (which need not be the same); and a list of Special Use Airspace (SUA) zones through which the aircraft may not fly. The primary objective is to find a flight plan that maximizes the summed priority of the observations of the observations performed. A secondary criteria is to maximize efficiency (the proportion of the flight spent performing observations). Since it is intractable to find the best possible plan, we limit ourselves to searching for *good* plans that perform many observations of high priority. Solving the SFPP requires selecting the set of observations to service, ordering them and inserting necessary dead-legs.

Constraints on Valid Flights

In this section we describe the constraints on valid solutions to the SFPP. The constraints linking aircraft motion and observation feasibility are the most important component of the problem, so we describe them in detail here. If an observation is scheduled, then it must be performed for the requested duration without interruption. As we will see, the elevation depends on the coordinates of the object being observed, the position of the aircraft, and the time. SOFIA can view objects between 20° and 60° of elevation; checking this constraint requires first computing the aircraft's ground track throughout the course of the observation. Figure 1 shows the interaction between the object's position in the sky at a particular time, the aircraft's ground track, and the telescope elevation. The Earth is modeled as an oblate spheroid $\mathbf{E}$, whose surface is defined by the equation $\frac{x^2}{a^2} + \frac{y^2}{a^2} + \frac{z^2}{c^2} = 1$ where $c < a$.

Let $\mathbf{p}$ be the aircraft's current position, (latitude γ and longitude L) and θ be the (Sidereal) time that the aircraft is at $\mathbf{p}$. Let $\vec{\mathbf{S}}$ be the vector from the center of $\mathbf{E}$ to $\mathbf{p}$. Let $\vec{\mathbf{T}}$ be the vector defining the vector to an astronomical object o, and $\mathbf{P}$ as the plane tangent to $\mathbf{E}$ at $\mathbf{p}$. Let $\hat{\mathbf{i}}, \hat{\mathbf{j}}, \hat{\mathbf{k}}$ be the unit vectors in the x, y, z directions respectively. Let $\vec{\mathbf{N}}$ be the vector normal to $\mathbf{P}$: $\vec{\mathbf{N}} = \frac{p_x}{a^2}\hat{\mathbf{i}} + \frac{p_y}{a^2}\hat{\mathbf{j}} + \frac{p_z}{c^2}\hat{\mathbf{k}}$ (Note that $\vec{\mathbf{S}}$ and $\vec{\mathbf{N}}$ are generally not parallel since $\mathbf{E}$ is a spheroid.) Let $\vec{\mathbf{T_P}}$ be the projection of $\vec{\mathbf{T}}$ onto $\mathbf{P}$; this is the *object azimuth* at $\mathbf{p}$, and is given by

$$\vec{\mathbf{T_P}} = \vec{\mathbf{T}} - \frac{\vec{\mathbf{T}}\vec{\mathbf{N}}}{||\vec{\mathbf{N}}||^2}\vec{\mathbf{N}} \quad (1)$$

Let $\vec{\mathbf{V}}$ be the desired heading of the aircraft. The observatory must track the object inducing $\vec{\mathbf{T}}$, subject to the constraint that the angle between $\vec{\mathbf{V}}$ and $\vec{\mathbf{T_P}}$ is 270°, because the telescope points out the left-hand side of the aircraft. Let $\mathbf{R}_{\vec{\mathbf{N}}}(\mathbf{270°})$ be a rotation matrix that rotates a vector 270° around $\vec{\mathbf{N}}$, and v be the airspeed of the aircraft; then

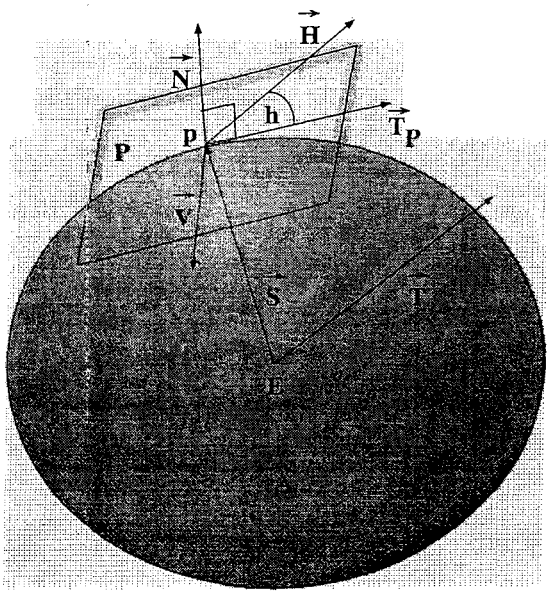

Figure 1: The Cartesian formulation of the instantaneous equations of motion of the aircraft and the elevation. We have exaggerated the spheroid E.

$$\vec{\mathbf{V}} = v\mathbf{R}_{\vec{\mathbf{N}}}(\mathbf{270°})\frac{\vec{\mathbf{T_P}}}{||\vec{\mathbf{T_P}}||} \quad (2)$$

Let $\vec{\mathbf{H}}$ be the elevation vector with respect to $\mathbf{P}$. We also require the angle h between $\vec{\mathbf{H}}$ and $\vec{\mathbf{T_P}}$ obey the constraint $20° \leq h \leq 60°$ throughout an observation. Most targets are sufficiently far from Earth that we can assume $\vec{\mathbf{H}} = \vec{\mathbf{T}} + \vec{\mathbf{S}}$. From vector calculus we then get the equation for the elevation h:

$$h = \cos^{-1}\left(\frac{\vec{\mathbf{H}}\vec{\mathbf{T_P}}}{||\vec{\mathbf{H}}||\,||\vec{\mathbf{T_P}}||}\right) \quad (3)$$

$\vec{\mathbf{T}}$ is a function of o and θ; this is because the Earth rotates on its axis. The vector $\vec{\mathbf{T}}$ traces a circle of radius $x^2 + y^2 = \frac{c^2-d}{c^2}$, where $d = |\frac{\delta}{90°}|$ in 24 hours (see (Meeus 1991) for an explanation of this).

The instantaneous change in $\mathbf{p}$ as the aircraft tracks o is $\frac{d\mathbf{p}}{d\theta} = \vec{\mathbf{V}}$. Since $\vec{\mathbf{V}}$ is a function of $\vec{\mathbf{T}}$, it is a function of $o, \mathbf{p}$ and θ. Solving for the ground track is necessary to compute h and check the elevation constraints. It is worth noting that this formulation also makes it easy to add the effect of winds by adding the appropriate vectors to $\vec{\mathbf{V}}$, and also correct for aircraft pitch by rotating about $\vec{\mathbf{V}} \times \vec{\mathbf{N}}$, but we omit these for brevity. The ground track and elevation constraints are solved using 5^{th}-order Runge-Kutta (Cash & Karp 1990) with error-adaptive step sizing.

The telescope is carried aboard a Boeing 747-SP aircraft. The fuel consumption of each engine depends on the aircraft weight, mach number, outside air temperature, initial altitude and final altitude. The fuel consumption constraints are represented in a lookup table provided by Boeing; space precludes describing the fuel consumption constraint in more detail. The aircraft follows a pre-determined *altitude profile* that describes the maximum permitted altitude at an absolute time after takeoff. Climbs are allowed periodically to increase engine performance; in addition, there is less water vapor at higher altitudes, and so better science is possible [1]. The combination of atmospheric conditions and aircraft weight may force the aircraft to fly lower than the altitude profile permits. At the end of a leg, if the aircraft is allowed to climb, it climbs to the maximum altitude permitted by the fuel performance table or the altitude profile. The profiles for SOFIA were developed originally in (Becklin & Horn 2001). Gridded wind and temperature climatology data is available to correct the ground track in the face of winds and provide data for calculating fuel consumption. Finally, SUAs constrain the ground track of the aircraft.

ForwardPlanner and its Discontents

ForwardPlanner, an algorithm for solving the SFPP, is described in in (Frank & Kürklü 2003) and (Frank, Gross, & Kürklü 2004). ForwardPlanner combines progression based search, heuristics and stochastic sampling, resulting in a fast, incomplete randomized algorithm. The core of the algorithm is observation feasibility checking. An observation is *feasible* if there is a sufficiently short dead-leg that makes the observation visible, the observation remains visible in darkness for long enough, the ground track of the dead leg and the flight leg do not cross any SUAs, and the aircraft can fly to the landing airport. We elaborate on this description as follows: suppose the aircraft is at position α, L at time θ. Consider the *shortest dead-leg* making an observation visible for long enough. If the resulting flight leg crosses any SUA, the observation is rejected. If the dead leg crosses an SUA, the heading is shifted minimally left or right from the heading of the shortest dead leg until the dead leg misses all SUAs. The duration of the leg is then adjusted to ensure the object is visible for the required duration. If the resulting dead leg is longer than D (an operational limitation on the longest permissible dead-leg), then the observation is rejected. If the observation begins before sunset or ends after sunrise *at the local position*, the observation is rejected. (Remember, changing your position changes the time at which the sun rises or sets.) Finally, if the aircraft cannot return to the landing airport after the observation is performed, the observation is rejected. If it survives all of these checks, it is feasible.

[1] Constraints on permitted water vapor for feasible observations are not treated in this paper.

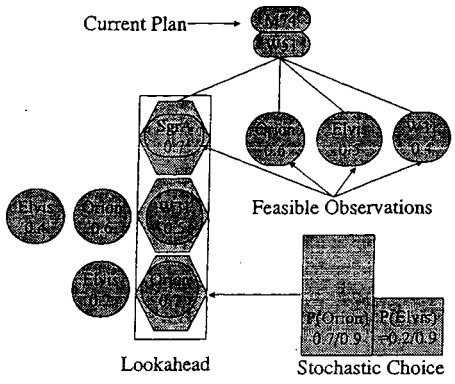

Figure 2: ForwardPlanner's search. At each step, all feasible observations are considered as the next observation in the plan. For each feasible observation, an extension of the plan is built using lookahead. The extensions are evaluated to determine which observation to perform next (the numbers inside each feasible observation.) The values are used to construct a probability distribution used to choose the next observation; each choice is indicated by a hexagon.

Each feasible observations is evaluated by first adding the observation to the flight plan, then heuristically adding a fixed number of additional observations. This "lookahead" is performed to estimate the best flight plan possible after adding each observation. These short extensions are evaluated using a weighted sum of the *priority* of the observations performed so far, the *efficiency* (ratio of time spent observing to total flight time) of the (incomplete) flight, the estimated time to return to the designated landing airport, and the total time spent in turns. The heuristic rank of each observation is treated as the mass of a probability distribution used to select the next observation. Thus, if we have a set of choices C and heuristic values of of these choices $v(c) \in C$, we choose an element $c \in C$ with probability $\frac{v(c)}{\sum_{d \in C} v(d)}$. This technique is similar to Heuristic Biased Stochastic Sampling (HBSS), a technique used for scheduling ground based telescopes (Bresina 1996). This means that the "best" candidate need not be selected at any stage of the process, but has the highest probability of being selected next. The process of evaluating the feasible observations and adding the next observation to a flight is shown in Figure 2.

The principal cost of ForwardPlanner is in the lookahead phase, where many legs are constructed to test observation feasibility. Let N be the number of observation requests, let K be the lookahead depth, and let M be the maximum number of observations that can be in any flight plan. Then ForwardPlanner makes $O(N^2KM)$ calls to Feasible(); a

proof of this appears in (Frank & Kürklü 2003). The calls to Feasible() include numerous dead-leg construction steps. The basic algorithm was improved upon in (Frank, Gross, & Kürklü 2004) by observing that many of these expensive dead-leg construction steps could be eliminated by formulating the problem of finding dead-legs to prove feasibility as a zero-finding problem. This approach reduced the runtime of ForwardPlanner without impacting the value of the flight plans found. Further reductions in runtime were later accomplished by approximating the flight dynamics calculations using Euler's Method, which involves flying a constant heading for a fixed (small) duration relative to the total observation time. Unfortunately, this approach lead to a reduction in the value of the plans found. Furthermore, as SUA and altitude sensitive fuel consumption constraints were added to the problem description, the Euler's Method approach failed to deliver good quality flight plans.

Squeaky Wheel Optimization for the SFPP

Squeaky Wheel Optimization (SWO) (Joslin & Clements 1999) was originally developed for scheduling problems with an optimization objective. SWO employs a permutation of tasks to schedule, and a fast procedure called a *Constructor* that treats each task in order, ultimately scheduling tasks or rejecting them. The permutation and its resulting schedule are then analyzed by a *Critic* to determine a new permutation that might schedule tasks that were previously rejected. The cycle repeats until all tasks are scheduled or for a fixed number of iterations. SWO was originally evaluated on Graph Coloring (Joslin & Clements 1999), and has since been employed for satellite observation scheduling (Globus *et al.* 2004) and range scheduling (Barbalescu, Whitley, & Howe 2004), as well as project scheduling with temporal constraints (Smith & Pyle 2004).

Figures 3 and 4 describe a version of SWO specialized for solving the SFPP. An initial permutation of the observations and takeoff time are generated. The Constructor assumes that the flight begins at the takeoff time, and that the permutation P imposes a precedence ordering on the observations, and attempts to construct a schedule. If any of the observations are rejected, SWO attempts to modify the permutation P by analyzing the rejected observations and the resulting flight plan F. There is a complex interplay between the permutation modification and takeoff time selection. First, it is possible to construct very bad flight plans by poor selection of the takeoff time. Second, the combination of the takeoff time, permutation and the fast scheduler implicitly schedules a subset of the observations. Finally, the fact that permutations are constantly modified allows reconsideration of the takeoff time based on the new permutation. For these reasons, this version of SWO ensures that new takeoff times can be chosen after each modification of the permutation.

In preparation for describing our SWO variants, we will introduce some concepts. The SFPP can be relaxed by throwing away the flight dynamics and fuel consumption, effectively pretending that the observatory is fixed at the takeoff airport. This leaves a problem in which observations have release times (earliest rise times), due dates (latest set times), occupy a unary resource (the telescope). Since SOFIA has a maximum elevation limit, the true feasibility

Feasible(o, p)
\# o is the observation
\# p is the current position
\# D is maximum dead leg duration
(d, h, z) = Newton(o,p)
\# h = heading, d = duration, z = SUA zone
if the dead-leg crosses any SUA zone z
  h' is heading s.t. all z not crossed minimizing $|h - h'|$
  d' is new duration s.t. visibility constraint satisfied
  $d = d'; h = h'$ \# Revise dead leg to avoid SUAs
  if $d > D$ **return false**
if observation starts and ends in darkness
  if dead leg home after dead-leg followed by o
    return true
return false

Figure 3: The feasibility test with Newton's Method for finding dead legs.

windows of objects may not be convex. Additionally, objects could set then rise during the night, but usually objects are observed at times of year when they are visible all night (and thus achieve their maximum elevation sometime during the night). This approximation is not bounding, because objects may rise earlier and set later at different positions than the takeoff airport. The approximation is still an $\mathcal{NP}$-Complete problem [2], but we use approximate solutions of this problem as heuristics.

Convex time windows during which an object at Right Ascension α and declination δ is visible can be constructed as follows. If the aircraft is at latitude γ longitude L, the earliest and latest times θ_r, θ_s at which the observation is visible by SOFIA are given by (Meeus 1991):

$$\theta = \cos^{-1}\left(\frac{\sin(20) - (\sin\delta)(\sin\gamma)}{(\cos\delta)(\cos\gamma)}\right) + L + \alpha$$

The $\sin(20)$ term arises from the fact that SOFIA's lower elevation limit is 20°. Note that $\cos^{-1}(x)$ has 2 solutions, which provide the earliest rise time θ_r and latest set time θ_s of the object at this position. The time of sunset and sunrise at this position can be used to further tighten this window. There can be at most 2 feasible windows; by default we construct the *first* feasible window. We will also often refer to the time at which an object reaches its maximum elevation (above the local horizon), called the *transit time*. This is simply $\frac{\theta_s+\theta_r}{2}$.

Constructing a Flight Plan

As Figure 4 indicates, construction of a flight plan occurs after generating an order of the observation requests and a takeoff time. Each observation in the permutation is tested for feasibility given the current position of the aircraft and the current time. Feasible observations are added to the flight plan unconditionally, while infeasible observations are rejected.

[2] According to Graham's hierarchy, it is $1||\sum w_i U_I$ which Karp proved $\mathcal{NP}$-complete (Brücker 1998)

```
SWO(MaxFlights,MaxRepeats)
# F is (initially empty) current flight plan
# B is (initially empty) best flight plan
# P is a permutation of the requested observations
# R is (initially empty) rejected observations
for MaxRepeats
    Generate permutation P
    for MaxFlights
        Select the takeoff time
        # Construct flight from P
        # p is the current position of F
        for observation o ∈ P
            if Feasible(o, p)
                Add p to F
            else add p to R
        end for
        Update best flight plan B
        if R = ∅ return F
        else # Modify P by analyzing F and R
            Choose object(s) in R to reorder
            Choose position(s) in P using F to reschedule
    end for
end for
return B
```

Figure 4: A sketch of the family of SWO-based Flight Planning Algorithm. Pseudocode lines in italics indicate design choices that will be described in later sections.

Generating Initial Permutations

Generating initial permutations of observations is performed in the following ways:

- **Uniform**. If there are N observations, one of the $N!$ permutations is chosen uniformly at random.
- Sort by Earliest Start Time **Rise** at the takeoff airport. We calculate θ_r as described in the previous section. The intuition behind this ordering is that flights often occupy the whole night, so beginning observations as early as possible is a good initial guess. Furthermore, this allows the largest time window to observe any object.
- Sort by Latest Start Time **Set** at the takeoff airport. We calculate θ_s as described in the previous section. Observing an object as late as possible may be a cheap method of ensuring enough time remains to schedule necessary dead-legs.
- Sort by Transit Time **Transit** at the (landing) airport. The intuition here is that this allows observing very nearby the airport; while one object is being observed, the next object moves closer to the landing airport, allowing the aircraft to "loiter" nearby.

Generating Takeoff Time

As we previously observed, due to the combination of oversubscription and the complex nature of the visibility constraints, choosing a good takeoff time is important to constructing good flight plans. We therefore break down the takeoff time generation process into 2 phases: determining a *range* of takeoff times, and *sampling* from the range.

Generating a takeoff time can be done as follows:

- Estimated flight duration **FlightDur**. If we simply assume that the aircraft will stay aloft as long as possible, we can estimate the flight duration from the initial fuel load and flight profile. The takeoff time range is sunset to sunrise minus this estimated flight duration. Since this quantity is independent of the permutation, it needs be calculated only once. Especially in the summertime for long flights, this approach will reduce the takeoff time range to one time (roughly half an hour before sunset).
- Minimum of Earliest Start Times **Min Rise**. We can calculate the minimum over all θ_r at the takeoff airport, and "pad" this by the amount of time needed to climb to operational altitude. Since this quantity is independent of the permutation, it needs be calculated only once. Only one takeoff time is generated by this approach.
- Optimize **First Observation** in Permutation. It is clear that θ_r can be a bounding above approximation to the earliest time when an observation can be performed; to see why, observe that flying towards the observation makes it possible to observe it earlier. Another approach is to assume that the first observation in a permutation is meant to be observed, and to calculate the earliest time at which this observation can be performed. Binary search over takeoff times is performed to find the takeoff time leading to the earliest feasible observation time for the first observation. Only one feasible takeoff time is generated by this approach. As the first observation in the permutation can change, the takeoff time will need to be recalculated.
- Approximate solution to the relaxed scheduling problem **Feas-Sched** . The takeoff time range calculation can use the current permutation as a guide to the actual schedule that will be constructed. Since all we are interested in is a range of takeoff times, we consider only fast approximate solutions to the relaxed problem. We use the θ_r and θ_s calculated at the takeoff airport to approximate the time windows for the observations. A feasible solution to the relaxed scheduling problem can be generated using the permutation as an ordering heuristic, and either greedily scheduling from the beginning or the end of the permutation. (It is trivial to see that different feasible schedules, and different takeoff time ranges, can be generated by scheduling forwards or backwards.) Once a feasible solution is generated, we calculate the slack of the first feasible observation, again "padding" for the time to climb to altitude.
- An elaboration **Feas-First** on the previous method is to adjust the θ_r at the takeoff airport based on optimizing the first observation's dead leg to observe the object as early as possible.

If a range of takeoff times is generated, we select from them uniformly at random.

Modifying the Permutation

In order to modify the permutation P, we must build a critic to both select a rejected observation r in R and decide where in P we would like to put r. We want to use the flight plan F built with permutation P to decide how to modify P. Each observation in a flight plan defines a "slot" in which a new observation could be placed. Unlike SWO approaches taken

in (Barbalescu, Whitley, & Howe 2004) and (Globus *et al.* 2004), we do not perform "blind" migration of jobs in the permutation. Rather, we identify where in the permutation we can move rejected observations to ensure that the resulting schedule is modified. Since we guarantee that a rejected observation will be scheduled during the next construction phase, we run the risk that some later observations in the flight might be displaced. Thus, it is important to estimate how much we "regret" moving an observation to a particular place.

We break critics up into phases. First off, we must determine which observations can be performed in each slot. Second, we must modify the permutation by choosing rejected observations to move and a slot for the observation to occupy, which implicitly determines where in the permutation the observation will move.

Feasibility testing is done as follows. For each rejected observation and candidate slot in the flight plan, the aircraft begins at the position and time defined by the prefix of the flight. The feasibility testing procedure of Figure 3 is used to determine feasibility. Regret calculations are done as follows. For each feasible observation-slot pair, we could run the constructor and look at the difference in the number (alternatively, summed priority) of observations scheduled by the constructor after modifying the permutation. Given the expense of the constructor, however, we *approximate* the "regretted" observations. This is done by examining the time at which the new observation ends. Since the new observation is guaranteed to be feasible, successive observations will be delayed, both due to the duration of the new observation and its dead leg (if any). We then evaluate the rate of change of the elevation of each successive observation to find out if it would still be visible at the same position at the later time. This is obviously an approximation, since the aircraft position would change after the newly inserted observation. Furthermore, it doesn't consider the possibility that unscheduled observations in the permutation could be added, so it is a conservative regret estimate. In what follows, assume the problem instance contains N observation requests. All of our critics use the biased sampling approach described earlier to make selections. Recall that if we have a set of choices C and values of of these choices $v(c) \in C$, we choose an element $c \in C$ with probability $\frac{v(c)}{\sum_{d \in C} v(d)}$.

Permutation modification can be done as follows.

- **2-Phase.** The feasibility calculation is performed first. The regret is calculated for each feasible observation-slot pair as follows. The first and last slots in a flight are special cases; if either the last or first slot is feasible, the heuristic value of the pair is N. If no observations in the plan are "regretted" given a possible move, the value of the pair is cN where c is a large constant. Otherwise, if a "regretted" observation had a dead-leg, the move is penalized by 0.5, otherwise it is penalized by 1. If r is the sum of the regret penalties for the pair, the value of the pair is $\frac{1}{r}$.
- **3-Phase.** Again, the feasibility calculation is performed first. In this case, however, we separate the choice of the observation and the slot.
  - Choose observation first. We have three ways to choose the observations. The first approach considers only the feasible slots **Obs-Feas** in the existing flight plan as the basis for choosing observations; the intuition is that this approach abstracts what the constructor does. When using this approach, we assign higher probability to observations with fewer slots available, on the assumption that these observations are "harder" to insert, and will result in worse flight plans if we don't try and fix them at this step. If an observation is visible in s slots, the heuristic is $N + 1 - s$. (Observations visible nowhere are not chosen.)

    We also have two "Time window" based options for observation sampling, which accounts for the available time when an observation can start. For each rejected observation flight plan, we first calculate θ_s and θ_r and choose the observation with the largest value of $\theta_s - \theta_r$. When θ_s and θ_r are calculated at the takeoff airport $\mathbf{Time}_t$, this calculation can be performed once and needs never be repeated. The heuristic value for each observation in this case is $\frac{\theta_u - \theta_l}{\theta_s - \theta_r}$. When θ_s and θ_r are calculated at each slot in the current flight $\mathbf{Time}_f$, we expect improvement, but at a higher computational cost. If S is the set of slots in the flight, the heuristic value of each observation is $\min_{s \in S} \frac{\theta_u - \theta_l}{\theta_s - \theta_r}_s$.
  - Choose slot first **Slot-Feas**. This approach also considers only the feasible slots in an existing flight plan; however, we choose the slot first. We assign higher probability to slots with fewer feasible observations, on the assumption that putting these observations somewhere else will reduce the possibilities for competing observations, resulting in worse flight plans later on. If v observations are visible in a slot, and the problem instance contains N observations the heuristic is $N + 1 - v$. (Slots with no visible observations are not chosen.)

  In the three phase critics, the first selection reduces the feasible observation-slot pairs. The second selection is made based on the regret of the remaining feasible observation-slot pairs, as described above.

As a final wrinkle, we can modify the permutation by moving k rejected objects rather than just one. The idea here is that multiple rejected observations could be re-ordered *independently* and potentially improve the flight plan using fewer construction steps. This idea was successfully employed by (Globus *et al.* 2004) and (Barbalescu, Whitley, & Howe 2004) to speed up SWO.

Identifying the Right SWO Features

As we previously stated, the most time-consuming feature of the SFPP is feasibility testing, which often involves many leg construction steps. Whereas ForwardPlanner makes $O(N^2KM)$ flight leg feasibility checks, SWO makes $O(N)$ feasibility checks; however, since one of the two parameters likely scales with N for good performance on larger problem sizes, this is misleading, and should be treated with caution.

One of the problems with designing algorithms such as SWO is that there are vast numbers of possible algorithms; a brief summary of the options for each feature appears in

Permutation	Takeoff Range	Critic	1^{st} Choice
Uniform	**FlightDur**	**2-Phase**	
Rise	**Min Rise**	**Obs-Slot**	**Time$_f$**
Set	**First Observation**	**Slot-Obs**	**Time$_t$**
Transit	**Feas-Sched (Fwd)**		**Obs-Feas**
	Feas-Sched (Bkwd)		**Slot-Feas**
	Feas-First (Fwd)		
	Feas-First (Bkwd)		

Figure 5: Summary of SWO Feature choices. Notice that not all nuances of each feature are in the table.

Figure 5. Furthermore, as described in the previous paragraph, some of the features are tightly coupled, and only a few of the combinations are likely to work well in practice. Finally, there is the matter of tuning the number of restarts. Our approach to this problem is to use a version of SWO with random selection of features as a baseline. This baseline algorithm is: **Flight-Duration** based takeoff time range selection, **Uniform** random initial permutation, and a **3-Phase** critic that first randomly chooses observations, then slots according to the **Time$_t$** rule, and then chooses slots based on approximate regret. We will then optimize takeoff time selection. In part, this is because it appears to be a very important component of the constructor on its own, and in part because good critics and good permutation selection appears to depend on it. We will then optimize permutation and critic dependent on the takeoff time. At all phases of testing, we will use the Wilcoxon Signed Ranked Test (Lindgren 1976) to determine whether each algorithmic variant is superior to the baseline SWO; we will select a small subset of promising algorithms to generate the next algorithm. In the presentation of the Wilcoxon test results, positive z indicates an algorithm variant is likely to perform better than the baseline, while a negative z indicates an algorithm variant is likely to perform worse than baseline. Criticality measurements are typically given in ranges; criticalities of > 0.05 are not considered statistically significant.

Empirical Results

In this section we present empirical results for varying facets of SWO in order to find the best overall algorithm for solving the SFPP.

Sample Problems

In Figure 6 we tabulate the number of observations, the archived flight duration, and the airport.

We used as a benchmark the problem instances described in (Frank & Kürklü 2003) to determine the utility of our new techniques. These instances are modified in a number of ways. Most importantly, the flight horizons are wider, and so takeoff time is between sunset and sunrise. The climatology data from European Center for Medium Range Weather Forecasting. [3] are used to provide outside air temperature for calculating fuel consumption. The initial fuel load is calculated from the climatology data for each day of flight, and is based on the altitude profile 4 from (Becklin & Horn 2001). This profile conforms to realistic expectations that good observing will require an altitude of at least 39000 ft,

[3] www.ecmwf.int

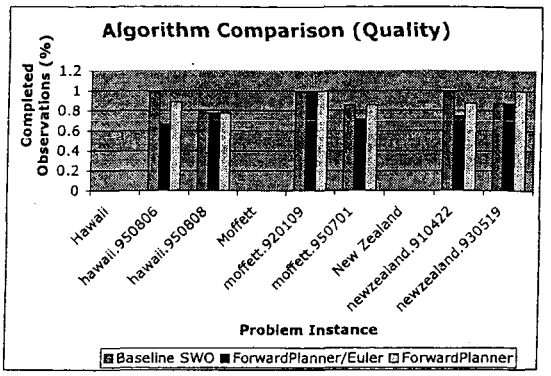

Figure 7: Comparison of solution quality for ForwardPlanner (with and without Euler's method approximation of flight dynamics) and SWO.

and leads to non-trivial takeoff time ranges requiring caution in selecting takeoff times, as well as introducing the option that not all observations can be scheduled. Finally, SUAs impact flights from Moffett and Hawaii; we use data from the National Geospatial Intelligence Agency's Digital Aeronautical Flight Information File.

The priorities of all observations are identical, and all observations can be scheduled on a designated flight day Thus, the principal goal is to find an efficient flight with all of the observations scheduled. The maximum dead-leg duration was set to 4 hours. For the dead-leg search using Newton's Method we used a step cutoff of 150 and error tolerance $t = 10^{-6}$. The step parameters used in forward differencing were: $s_1 = 0.01°$ and $s_2 = 60$ seconds. When CPU times are reported, these experiments were run on a Sun Workstation with dual 600 MHz CPUs and 2048 Mb memory. Unless otherwise stated, MaxFlights = 20 and MaxRepeats = 10.

Figures 7 and 8 compare performance on 6 sample problems. We see that Euler's Method saves considerable time in ForwardPlanner, but leads to plans with fewer scheduled observations (with one exception). The baseline SWO provides plans of as good or better quality as ForwardPlanner, but at a fraction of the time of ForwardPlanner with the Euler's Method approximation speedup.

Choosing Takeoff Times

The results of varying the takeoff time selection while holding all other aspects of SWO the same are shown in Figure 9. In this figure we present the Wilcoxon Ranked Sign test output for the best percentage of the observations found by SWO. Again, recall that we compare each new SWO variant to the baseline SWO described in the previous section. As

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Airport	H	H	H	H	M	M	M	M	M	M	M	M	M	M	M	M
Date	8/6	8/8	8/10	8/12	1/9	1/10	1/16	6/16	6/18	6/19	6/30	7/6	8/12	8/16	4/4	4/5
# Obs	9	9	10	10	7	8	8	6	10	8	8	6	11	10	9	9
Index	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Airport M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Date	4/6	4/11	4/12	4/14	4/19	5/4	5/8	7/1	7/6	8/2	8/22	8/24	8/26	8/29	9/1	9/19
# Obs	10	8	8	8	10	10	6	7	4	6	9	8	11	10	8	7
Index	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	
Airport	M	M	M	M	M	M	M	MH	MH	MH	N	N	N	N	N	
Date	9/20	9/21	9/23	9/26	9/28	9/29	10/4	6/21	7/12	8/4	11/25	4/22	5/11	5/15	5/19	
Obs	7	3	10	8	8	8	4	8	7	7	10	8	8	8	8	

Figure 6: Characteristics of Single Day Instances.

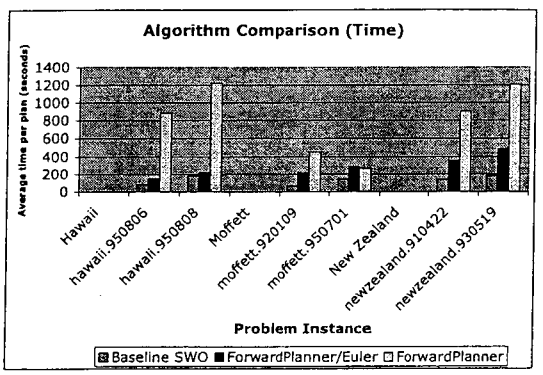

Figure 8: Comparison of average CPU time for ForwardPlanner (with and without Euler's method approximation of flight dynamics) and SWO.

Takeoff Range	N	Z	Crit.
Min Rise	17	-2.21823	[0.01,0.025]
First Observation	18	-2.05775	[0.01,0.025]
Feas-Sched (Fwd)	12	1.31398	>0.05
Feas-Sched (Bkwd)	16	1.82273	[0.025,0.05]
Feas-First (Fwd)	10	2.31889	≈ 0.01
Feas-First (Bkwd)	9	2.2131	≈ 0.01

Figure 9: Wilcoxon Ranked Sign Test results comparing SWO Takeoff Time variants to SWO Baseline.

Takeoff Range	Permutation	N	Z	Crit.
Feas-Sched (Fwd)	**Random**	12	1.31398	>0.05
Feas-Sched (Fwd)	**Rise**	14	1.61649	≈ 0.05
Feas-Sched (Fwd)	**Set**	12	1.3532	> 0.05
Feas-Sched (Fwd)	**Transit**	12	1.43165	> 0.05
Feas-Sched (Bkwd)	**Random**	16	1.82273	[0.025,0.05]
Feas-Sched (Bkwd)	**Rise**	14	2.05593	[0.01,0.025]
Feas-Sched (Bkwd)	**Set**	14	1.42816	> 0.05
Feas-sched (Bkwd)	**Transit**	14	1.49094	> 0.05

Figure 10: Wilcoxon Ranked Sign Test results comparing SWO Initial Permutation ordering variants to SWO Baseline.

we can see, the least "informed" approach, **Min Rise**, performs worst. Optimizing the takeoff time range of the first observation also did not perform well. Both of these approaches perform worse than the baseline SWO. Solving the relaxed feasible scheduling problem to generate the takeoff time range did well but appeared to take more CPU time. Interestingly enough, for one problem instance using θ_r and θ_s calculated at the takeoff airport performed worse than optimizing the takeoff time of the first observation. This is because an SUA constraint interacts with the visibility constraint on the first observation, forcing it happen at a different time; no observations were scheduled for this problem. Optimizing the takeoff time of the first observation captures this dependence, and leads to much more takeoff time flexibility. This one flight leads to the large difference between the **Feas-Sched** and **Feas-First** approaches.

Generating Initial Permutations

For this series of tests, we tested both **Feas-Sched (Fwd)** and **Feas-Sched (Bkwd)** variants of takeoff time selection with the different initial permutation methods. The results of varying the permutation selection for these two takeoff time selection options while using the baseline critic are shown in Figure 10. Notice that **Random** is our baseline permutation method, and thus the first and fifth lines of Figure 10 are repeated from table 9.

The results of this test are also quite conclusive. Initially sorting observations using **Rise** is clearly an improvement over the baseline SWO, while the other initial permutation generation methods are not a clear improvement. It seems reasonable that rise-time based permutations should do well,

Critic	1^{st} Choice	N	Z	Crit.
3-Phase	Time_t	14	2.05593	[0.01,0.025]
3-Phase	Time_f	16	2.21055	[0.01 0.025]
3-Phase	Obs-Feas.	14	1.71066	[0.025 0.05]
3-Phase	Slot-Feas.	16	1.64175	>0.05
2-Phase	N/A	14	2.33842	[0.005 0.01]

Figure 11: Wilcoxon Ranked Sign Test results comparing SWO Critic variants to SWO Baseline.

given that the observations were chosen to be in the sky most of the night on the day of observing; it isn't so clear why **Transit** did not do so well. Interestingly, using takeoff time generation method **Feas-Sched (Bkwd)** is clearly a better complement to rise-time based initial permutation selection than **Feas-Sched (Fwd)**; we do not have a good explanation for this.

Modifying Permutations

For this series of tests, we tested the **Feas-Sched (Bkwd)** takeoff time generation method and **Rise** initial permutation generation method with each critic method. In each case, only one rejected observation was moved per critic application. The results of varying the critics are shown in Figure 11. Notice that $\textbf{Time}_t$ is our baseline critic method, and thus the first line of Figure 11 is repeated from table 10.

As expected, **2-Phase** is quite good. Also as expected, for **3-Phase**, we see that $\textbf{Time}_t$ is not as good as $\textbf{Time}_f$. Somewhat surprisingly, though, neither of the 3-Phase approaches using the exact feasibility calculation with sampling biased by the number of slots is very promising. Sampling observations first is a little better than sampling slots first, but neither appear as likely to improve over the baseline as the other methods tested. This suggests that even crude estimates of time are important when building the critics, and demonstrates that simply using slot counts is not good enough.

Our final critic experiments use **Feas-Sched (Bkwd)** takeoff time generation, **Rise** based initial permutation selection, and **2-Phase** critic. In this experiment we vary the number of rejected observations that are moved. The regret values are still used to sample, and are renormalized between samples. The number of observations is moderately low, so we limited ourselves to experiments moving 2, 3 or all rejected observations. As we see, it appears that we benefit from increasing the number of rejected observations that are moved; we conclude that the independence assumption is justified, and that multiple observation moves do not (unduly) interfere with eachother. However, there may be a quality penalty in increasing the number of rejects moved, as the difference is most pronounced only if we move all of the rejected observations.

The Best Algorithms

It remains now to compare the CPU performance of the SWO algorithms. In order to make sense of this analysis, it is important to note that SWO terminates a loop if all observations are scheduled. For the purposes of this presentation,

Rejects	N	Z	Crit.
1	14	2.33842	≈ 0.01
2	13	2.14898	[0.01, 0.025]
3	13	2.25381	[0.01, 0.025]
all	15	2.54163	≈ 0.005

Figure 12: Wilcoxon Ranked Sign Test results comparing critics moving variable numbers of rejected observations to SWO Baseline.

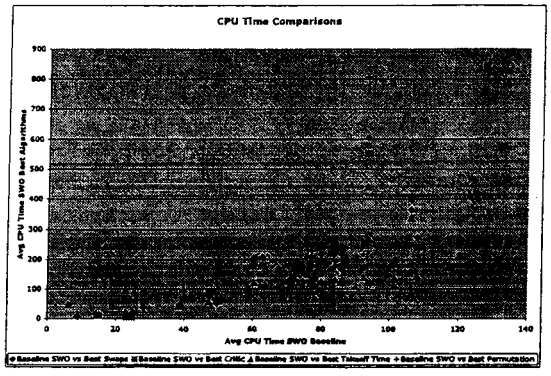

Figure 13: Comparison of baseline SWO CPU time with "incremental best" SWO variants identifying best SWO features.

we consider the "best" takeoff selection routine to be **Feas-Sched (Bkwd)** since **Feas-First (Bkwd)** is more costly and for various reasons was not used in the subsequent experiments. We will compare algorithm performance using the mean and variance in CPU times for all 20 runs of the different algorithms; CPU times are given in seconds. We also reproduce the Wilcoxon signed rank test results. The best SWO algorithm contained only one real surprise, which is the fact that the **Feas-Sched (Bkwd)** takeoff range generator appears best when coupled with the rest of the features. The takeoff time selection method imposes a significant computational burden on SWO, as can be seen by the increase in the mean CPU time. While the critics also impose a computational burden on SWO, we actually see a *reduction* in CPU time compared to those methods without the intelligent critics.

Name	Baseline	T/O	Perm.	Critic	Swaps
Takeoff Range	FlightDur	⇒	⇒	⇒	⇒
Permutation	Uniform	Rise	⇒	⇒	⇒
Critic	3-Phase, Rise_t	⇒	2-Phase	⇒	⇒
Swaps	1	1	1	all	
Mean	63.728	113.071	187.612	166.486	145.501
Sdev	29.976	77.623	144.755	108.985	86.427
N	-	16	14	14	15
Z	-	1.82337	2.05593	2.33842	2.54163
Crit	-	[0.025,0.05]	[0.01,0.025]	[0.005,0.01]	≈ 0.005

Figure 14: Comparison of mean and variance of SWO CPU times for all "incremental best" SWO variants identifying best SWO features.

We also show scatterplots of the average CPU time over all 20 runs of the SWO algorithm on an instance-by-instance basis in Figure 13. While this comparison technique ignores cases where one algorithm outperforms another on particular problem instances, it provides a more holistic view of the time comparison. We show four scatterplots comparing the CPU time of each of the best "incremental" SWO algorithm found with the baseline SWO. Our worst-case performance hit is roughly a factor of 5 increase in CPU time between the baseline SWO and the best SWO, which is moderately high; however, this leads to algorithms that are very favorable when compared to the previous approaches.

Conclusions and Future Work

We have described the application of SWO to the SFPP problem. Early results indicated that SWO was a powerful technique with the promise of delivering higher quality flight plans in less time than ForwardPlanner, our previous approach to the SFPP. A combination of approaches from ForwardPlanner, combined with analysis of SWO features, resulted in an SWO algorithm that fulfills this early promise. Takeoff time selection proved to be an important component of the overall approach. We also feel that we benefitted from "guided" critics that guaranteed permutation modifications resulting in new schedules.

There are several classes of future work to consider.

The next important feature of the SFPP problem involves water vapor constraints. Initial results with variants of SWO handling these constraints are promising, but expensive. Furthermore, there is a close relationship between fuel load, altitude and water vapor; full integration of this relationship into SWO will require more work. The SFPP also involves observation priority. Our experiments assumed all observations were of equal value; it is easy to generalize our SWO to handle variable priority, but we have no empirical results on performance. However, frequently astronomers want all of best observations, then all of second best, and so on. We have considered a "stratified" SWO that maintains one permutation per priority class. Controlling expense of this type of SWO and revising critics an interesting open issue. The SFPP also involves "hard constraints" for so-called "calibrators" that ensure that data from unknown observations can be properly analyzed. Such observations can be considered "top priority" objects and integrated into a "stratified" SWO. Finally, the SFPP also requires that we build series of flights rather than just a single flight. Preliminary flight series testing indicates that SWO is a promising technique for building flight series. However, more work is needed.

There are numerous options for elaborating on all aspects of our proposed SWO features, particularly critics. Examples include different sampling methods, mixing greedy selection with biased sampling, solving the relaxed scheduling problem for critics, using Euler's approximation in critics, and so on. Another interesting avenue of work involves employing mixtures of critics, and scheduling them at different "phases" of the SWO process. For example, observations incurring big dead-legs could be moved to try and increase efficiency. Also, observations early in flight will generally have worse water vapor (lower altitudes); "sensitive" observations pushed later in flight.

Acknowledgments

We would like to thank European Center for Medium Range Weather Forecasting for the use of the climatology data, Michael A. K. Gross for his ongoing assistance in this project, and Tien Ba Dinh for prototyping SWO for the SFPP. This work was funded by the SOFIA Projects Office and by the NASA Intelligent Systems Program.

References

Barbalescu, L.; Whitley, D.; and Howe, A. 2004. Leap before you look: An effective strategy in an oversubscribed scheduling problem. In *Proceedings of the* 19^{th} *National Conference on Artificial Intelligence.*

Becklin, E., and Horn, J. 2001. High-latitude observations on sofia. *Publications of the Astronomical Society of the Pacific* 113(786).

Bresina, J. 1996. Heuristic-biased stochastic sampling. In *Proceedings of the 13th National Conference on Artificial Intelligence.*

Brücker, P. 1998. *Scheduling Algorithms.* Springer.

Cash, J. R., and Karp, A. H. 1990. A variable order runge-kutta method for initial value problems with rapidly varying right hand sides. *ACM Transactions on Mathematical Software* 16:201–222.

Frank, J., and Kürklü, E. 2003. Sofia's choice: Scheduling observations for an airborne observatory. In *Proceedings of the* 13^{th} *International Conference on Automated Planning and Scheduling.*

Frank, J.; Gross, M. A. K.; and Kürklü, E. 2004. Sofia's choice: An ai approach to scheduling airborne astronomy observations. In *Proceedings of the* 16^{th} *Conference on Innovative Applications of Artificial Intelligence.*

Globus, A.; Crawford, J.; Lohn, J.; and Pryor, A. 2004. A comparison of techniques for scheduling earth observing satellites. In *Proceedings of the* 16^{th} *Conference on the Innovative Applications of Artificial Intelligence.*

Johnston, M., and Miller, G. 1994. Spike: Intelligent scheduling of the hubble space telescope. In Zweben, M., and Fox, M., eds., *Intelligent Scheduling.* Morgan Kaufmann Publishers.

Joslin, D., and Clements, D. 1999. Squeaky wheel optimization. *Journal of Artificial Intelligence Research* 10:353 – 373.

Lindgren, B. 1976. *Statistical Theory.* Macmillian.

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

Potter, W., and Gasch, J. 1998. A photo album of earth: Scheduling landsat 7 mission daily activities. In *Proceedings of the International Symposium Space Mission Operations and Ground Data Systems.*

Smith, T., and Pyle, J. 2004. An effective algorithm for project scheduling with arbitrary temporal constraints. In *Proceedings of the* 19^{th} *National Conference on Artificial Intelligence.*

Smith, D. 2004. Choosing objectives in over-subscription planning. *Proceedings of the* 14^{th} *International Conference on Automated Planning and Scheduling.*