[illegible]

SOFIA's Choice: Automating the Scheduling of Airborne Observations

Jeremy Frank

Caelum Research Corp.
NASA Ames Research Center
Mail Stop N269-1
Moffett Field, CA 94035-1000

1 Introduction

This paper describes the problem of scheduling observations for an airborne telescope. Given a set of prioritized observations to choose from, and a wide range of complex constraints governing legitimate choices and orderings, how can we efficiently and effectively create a valid flight plan which supports high priority observations?

This problem is quite different from scheduling problems which are routinely solved automatically in industry. For instance, the problem requires making choices which lead to other choices later, and contains many interacting complex constraints over both discrete and continuous variables. Furthermore, new types of constraints may be added as the fundamental problem changes. As a result of these features, this problem cannot be solved by traditional scheduling techniques. The problem resembles other problems in NASA and industry, from observation scheduling for rovers and other science instruments to vehicle routing

The remainder of the paper is organized as follows. In §2 we describe the observatory in order to provide some background. In §3 we describe the problem of scheduling a single flight. In §4 we compare flight planning and other scheduling problems and argue that traditional techniques are not sufficient to solve this problem. We also mention similar complex scheduling problems which may benefit from efforts to solve this problem. In §5 we describe an approach for solving this problem based on research into a similar problem, that of scheduling observations for a space-borne probe. In §6 we discuss extensions of the flight planning problem as well as other problems which are similar to flight planning. In §7 we conclude and discuss future work.

2 SOFIA: The Observatory

The Stratospheric Observatory for Infrared Astronomy (SOFIA) is an airborne observatory and ground support facility which will enable astronomers to perform infrared astronomy using an airborne platform. The observatory consists of a 747-SP aircraft with a 2.5m telescope. The telescope has an elevation range of 20-60 degrees, but only 2 degrees of horizontal freedom. The aircraft can fly with one of several instruments on board, enabling a wide range of science observations in the infra-red spectrum, between 0.3 to 1600 μ. SOFIA follows the Kuiper Airborne Observatory (KAO), which performed airborne astronomical observations for 20 years. More details on SOFIA can be found in [Bec97] and [ED97].

The operational goals for SOFIA are to fly 140 flights per year (3-4 flights per week), for as much as 8-9 hours at a time. These flights will be divided between Principal Investigators (PIs) and General Investigators (GIs). PIs will propose observations which occupy an entire flight, and will be responsible for their own observation and flight planning. GIs, on the other hand, will propose observations which may only require a small part of a flight, with the result that GI flights will perform observations requested by many different astronomers. These flights will be planned and performed by SOFIA staff.

This operational breakdown means that SOFIA staff will be faced with choices regarding how to plan flights in support of GI observations. Flight plans for the KAO were synthesized by hand, with the aid of software which computed trajectories from observation requests. The scope of the flight planning problem for supporting GI observations makes this approach to flight planning inadequate, as it would require too many person-hours to effectively schedule flights.

3 Planning for a Single Flight

In this paper, we advocate automating the process of flight planning in order to successfully meet the challenges of scheduling GI observations. There has been considerable success in automating the scheduling of jobs in a wide variety of industries with many different types of constraints. However, these problems are typified by relatively simple, homogeneous constraints, and the successful approaches depend on these simple representations.

The elementary problem for an airborne observatory like SOFIA is the *Single Flight Planning Problem* (SFP). This problem consists of constructing a good flight plan for a single flight on a given day. The problem input consists of the set of *observations* that have been requested, the constraints peculiar to the *flight environment*, and the *objective function*. As we shall see, this problem is too complex to be solved using traditional scheduling techniques.

3.1 The Observation Requests

An observation request consists of the name of the object to be observed, the amount of time requested, the relative importance of the observation, and a set of constraints on the observation. For now, we assume that the amount of time is fixed and also that it is strictly less than the maximum duration of the flight. The importance or priority of the observation is a summary of several different factors. Some observations are naturally more interesting to the science community than others. However, due to the limited duration of flights, it may be necessary to observe a target many times, and so it may be more important to finish a sequence of observations on a target than to start a new observation.

The most complex part of an observation request are the constraints on the observation. Some of these constraints are explicitly given by astronomers, while others are implicit, due to the nature of airborne observing. We now turn our attention to these constraints.

3.1.1 Ordering Constraints

Some observations may have explicit constraints on the order in which they are performed. For example, instruments may need to be calibrated by observing particular objects before the primary observation of interest is performed. In addition, the telescope may need to be tuned at the beginning and periodically during the flight by observing objects with particular characteristics. High-precision tuning may require observing the same object at multiple elevations, for instance. These requirements impose ordering constraints on the observations that must be obeyed [1].

3.1.2 Astronomical Constraints

Some objects may only be visible from certain positions on the earth at certain times of day. Thus, there may be an earliest start time and a latest end time for completing a given observation request. Astronomers may also provide explicit constraints on particular observations so that the data is of high quality. For example, the astronomer may require that the object be sufficiently far away from the moon or the sun, or that airmass or atmospheric water vapor is below a certain threshold. These constraints also dictate when a target may be observed. In particular, minimizing airmass requires

[1]While calibrations and setup operations are not strictly observations, for simplicity we represent them as such.

observing at a higher altitude, and minimizing water vapor can be accomplished by observing at higher altitudes or by observing further north [HB00].

3.1.3 Aircraft Constraints

SOFIA has complex constraints simply because it is an airborne observatory. Most objects appear to move through the sky as time passes. Because the telescope has little horizontal flexibility, the aircraft must fly a curved trajectory in order to keep objects in view. The wind speed, aircraft speed, and time and position an observation is started dictate the trajectory and final location of the aircraft at the end of the observation. Object visibility windows are further constrained by the limits on the telescope's angle of elevation. Even though an object may be visible from the ground, it may not be visible from the aircraft, either because it is too low or too high for the telescope to view. An object may sometimes have multiple windows of visibility during a single flight. For example, it may pass above and then below the maximum telescope elevation, requiring a choice of when to observe.

The aircraft must normally return to the airport it took off from. Flight time is also limited by fuel, requiring all observations to be done with enough time for the aircraft to return, from wherever it is, to the airport. Finally, the aircraft's altitude is constrained by its weight, but the weight decreases over time as fuel is consumed, so the aircraft can generally climb 2000 ft every two hours. These factors can interact with constraints on airmass or water vapor to further limit the windows during which observations can be made.

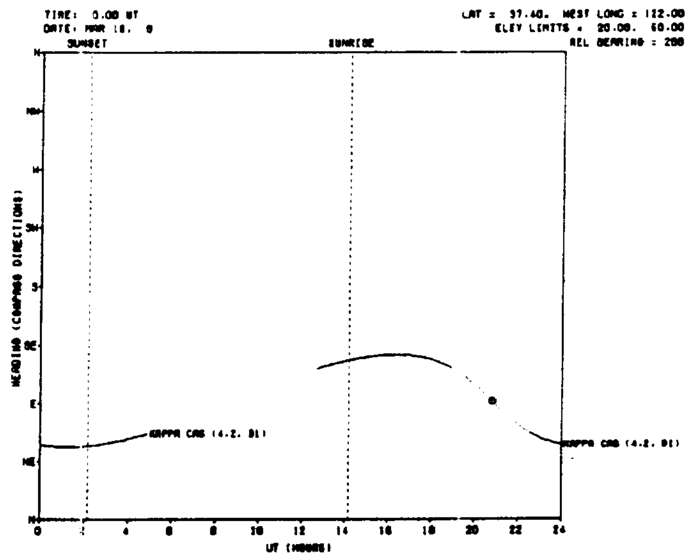

Figure 1: Visibility of an object during March 18, 2000 from Moffett Field. Note that time is given in Universal Time, and that sunset and sunrise are marked.

Many of these constraints are demonstrated in Figure 1. This figure shows visibility and heading information for an object viewed from Moffett Field during March 18, 2000. The y axis shows the heading the aircraft must fly to keep the object in view. The direction changes over time, indicating that the aircraft must constantly turn to keep the object in view. The curves on the plot indicate when the object is in view. No curve indicates the object is below the telescope's 20 degree minimum elevation, while the dotted curve indicates the object is above the 60 degree maximum elevation. Notice that the object passes below the minimum elevation and then returns to view then passes above the maximum elevation and again returns to view. During any 9 hour period, there are at most two windows of visibility for this object.

3.2 Flight Environment Constraints

In this section we discuss constraints derived from the environment on the day of the flight. One important example of such constraints are airborne warning zones, commercial flight routes, and other administrative restrictions on where the aircraft can fly [2]. Some flight environments may have fewer such restrictions; for example, if the aircraft flies out of Hawaii or New Zealand, there will be fewer such restrictions than flights over Nevada. Bad weather may constrain observations as well. While cloud cover is usually not an issue at the altitudes where observing is likely to occur, turbulence can affect the performance of the observations, and may increase observation time or have other effects on flights. Wind speed and direction can also have an effect on a particular flight. The aircraft's ground speed is directly affected by wind, and wind patterns change over time. Flight planners must take these effects into account when doing planning.

3.3 The Objective Function

The final component of the SFP is the objective function, which is used to compare two candidate flight plans. Within the confines of the single flight planning problem, the objective function can range in complexity. A good flight contains as many high-priority observations as possible; hence a good objective function might be to simply sum the priorities of the observations which are performed. However, the aircraft might spend considerable time flying without observing, i.e. flying a *dead leg*. Thus, the objective function may penalize dead legs explicitly. Astronomers may also *prefer* rather than require that observations be done at various water vapor levels, that targets be observed when they are far from the moon or other heavenly bodies, and so on. All of these preferences can then be added to the objective function, resulting in a fairly complex measurement of the goodness of a flight plan.

3.4 The Statement of the Problem

With all the components in place, we can now state the SFP: given a set of observations to perform, a date to perform them, a description of the environment on that date, and the objective function, select a (possibly proper) subset of the observations, a start time for each observation, a takeoff time, and specify any dead-legs. The resulting flight plan should maximize the objective function and must not violate any of the constraints. Figure 2 shows an example of a flight plan.

4 The Complexity of Flight Planning

Many efficient scheduling techniques rely on special encodings of problems in order to deliver good computational results. These techniques work only for simple constraints, such as equality and inequality or simple combinations of resource and precedence constraints. For example, many scheduling problems can be posed as linear programming problems, which can be solved in polynomial time [Lue84]. Many other examples of clever encodings and algorithms exploiting them can be found in a recent text on scheduling techniques [Bru98]. While these techniques are very powerful, the problems they can solve are limited, and a great deal of sophisticated modeling may be necessary to pose these problems in the correct form.

As we have seen, any instance of the SFP is composed of a large number of complex, heterogeneous constraints over both continuous and discrete variables. Even relatively simple versions of the SFP are quite complex, consisting of geometric constraints, precedence constraints, mutual exclusion constraints and temporal constraints, all in the same problem. While it may be possible to encode parts of the problem in order to take advantage of efficient algorithms, it is unlikely that we can find a good encoding which will serve for all instances of the problem. Furthermore, over the lifetime of the observatory, other constraints may be required to represent a flight planning problem. For example, new instruments may

[2]The KAO was not FAA certified, which meant that trans-national flights required extensive paperwork. SOFIA will be FAA certified and will not have these restrictions.

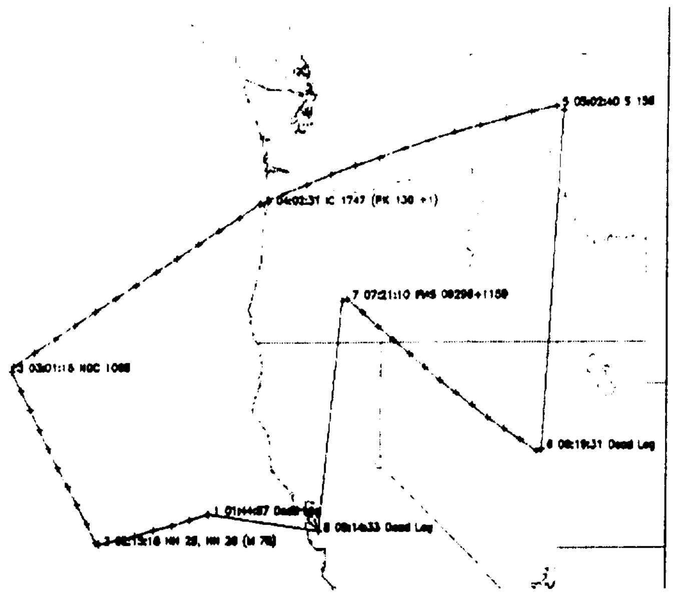

Figure 2: A flight plan. Each observation leg is marked with +s and labeled, while dead legs are simple lines. Note that no restricted areas are indicated in this figure; this flight plan is likely to cross restricted areas over Nevada.

have new constraints, and newly discovered objects may impose new constraints as well. The addition of these constraints may invalidate any overly specialized representations and algorithms.

The SFP presents another challenge in the form of the size of the representation. In particular, it is not known beforehand how many observations will be performed on any given flight, nor is it known how many dead-legs will be required. Each choice made during the planning and scheduling of the flight affects other choices later; for instance, choosing one set of observations makes it impossible to choose others. Traditional scheduling techniques require *all* of these choices to be represented, along with constraints which are triggered once the choice is made. Encoding all of this information results in large representations; for an extreme example, we refer the reader to [KS96]. The problem only becomes worse as the scope of the problem grows; thus, while it may be possible to encode a single SFP, it may not be possible to encode an extended SFP because the encoding will not fit into the computer's memory.

5 Automatic Generation of Flight Plans

In this section we describe a planning and scheduling paradigm which meets the requirements for solving the SFP. This paradigm has been successfully implemented in the New Remote Agent Planner [JMMR99], which is designed for use in space applications. We first discuss Dynamic Constraint Satisfaction Problems, a general representation for problems like the SFP. We then describe procedural constraints, which generalize constraints from mathematical relations to powerful, miniature programs. We then describe an algorithm which uses these procedures to solve DCSPs.

5.1 Representation

Constraint Satisfaction Problems or CSPs are a general representation which can be used to represent many problems, including scheduling problems. A CSP consists of a set of variables, each of which has an associated domain of legal values it can take on in a solution. In addition to variables, the CSP contains a set of constraints, which restrict the legal assignments to sets of values. These constraints can be extensional, in which all the legal assignments are listed, or intentional, in which legal assignments are encoded as a simple mathematical relationship. A constraint is *satisfied* if its variables are assigned values which are permitted by the constraint. A *solution* to a CSP is an assignment of each variable to a single value in its domain such that all the constraints are satisfied.

The notion of a constraint is very general, and many real-world problems can be represented very naturally with simple constraints. For example, [BF98] and [Nui94] both discuss various issues in representing scheduling problems as CSPs. However, sometimes representations of problems using simple constraints can become large and unwieldy. For instance, an enormous amount of space is required to explicitly represent all of the possible time-location pairs when an object is visible. It is often more efficient to represent a constraint with a mathematical relation such as $\leq$. *Procedural constraints* [J96] generalizes this concept by formalizing the notion of a procedure which enforces a relation among the variables of a constraint. A special form of procedural constraint called an *elimination procedure* is permitted to get rid of any element of a domain that is provably not part of any solution to the problem, given the information currently at hand. For example, an elimination procedure might eliminate all observations as candidates for the next observation on a flight because the aircraft is almost out of fuel. Procedures also are used to formalize the concept of *decision variables.* If a procedure is able to assign values to a set of variables $V - D$ given that all the variables in D have been assigned, then there is no reason to search the values of variables in $V - D$. The variables in D are called the decision variables, since those are the only variables over which search is performed. Continuous variables can be handled by making sure they are not in the set of decision variables [JF99].

CSPs are capable of representing a large number of interesting problems. However, CSPs contain no mechanism to express a preference between two solutions that satisfy all of the constraints. A *Constraint Optimization Problem* or COP is a CSP which includes a mapping from a solution to the real numbers. This mapping encodes the preferences between solutions which satisfy all the constraints. It should be clear from the discussion of the SFP above that we can pose the SFP as a COP.

As mentioned above, representing the SFP may be unwieldy due to the large number of constraints required to encode the conditional effects of all of the choices. An alternative representation is the *Dynamic Constraint Satisfaction Problem* or DCSP. A DCSP is a sequence of CSPs, in which each CSP is a modification of the previous CSP in the sequence. A CSP C is said to be a *relaxation* of a CSP D if C has fewer constraints, fewer variables or more combinations of assignments permitted in its constraints. A CSP C is said to be a *restriction* of a CSP D if C has more constraints, more variables or fewer combinations of assignments permitted in it's constraints. These ideas are formalized in [JF99]. DCSPs provide a way to formally characterize how a problem changes over time, and require less space since the impact of choices need not be encoded in a single representation of the problem.

5.2 Representing the SFP as a DCSP

In this section we describe how to represent the SFP as a DCSP. Let us assume that the problem consists of a set of observation requests, and our task is to construct a flight for a particular day such that the sum of the priorities of the observations performed exceeds a certain threshold. The durations of the observations are fixed, but we will permit dead legs in this problem. For simplicity, we ignore restricted zone constraints and artificially restrict the bearings of dead legs to the 4 cardinal directions, and restrict dead leg flight duration to 5, 10, 15 or 20 minutes.

The variables for the problem will represent aspects of each flight leg. Every leg will have a duration variable, and variables for the initial and final times and locations of the leg. Every observation leg also has an object and a priority variable, while every dead leg has a bearing variable. The constraints include those mentioned in §3, such as those imposed by astronomers, and by the problem instance itself. For instance, the constraint on the final ground location for an observation leg relates the initial and final locations and times, the duration of the observation, and the celestial coordinates of the object being observed. The constraint on the flight plan quality states that the sum of the priority variables, however many there are, must exceed a constant. Similarly, the constraint on the flight plan duration says that the sum of the leg durations, however many there are, must not exceed a constant. We also require that no two consecutive legs can be dead legs.

At any given instant in the construction of the schedule, we have a CSP consisting of the variables for the current set of legs and the constraints on those variables. However, it is not known beforehand how many legs a particular flight will require. We do know, however, that the flight must consist of at least 2 legs: one to take off from the airport and one to land. Thus, the flight will initially consist of two legs; the first will begin at the airport, the second will end at the airport. We must represent steps taken in flight planning which add new flight legs to the flight plan. This is modeled using the following variables and constraints. Each leg has associated with it two additional variables: *Leg-x* and *Leg-after-x*. There are three legal values for these variables: *dead-leg,obs-leg,home-leg*. If variable *Leg-x* is assigned the value *home-leg*, then the leg is connected to the leg which takes the aircraft home; otherwise, a new leg of the appropriate type is instantiated. We also add constraints requiring that the *Leg-x* variable can't be *dead-leg* or *home-leg* for observation legs, and that *Leg-after-x = Leg-x+1*. Since consecutive dead legs are prohibited, we add a constraint which prohibits both *Leg-x+1* and *Leg-after-x+1* from being *dead-leg* or *home-leg*.

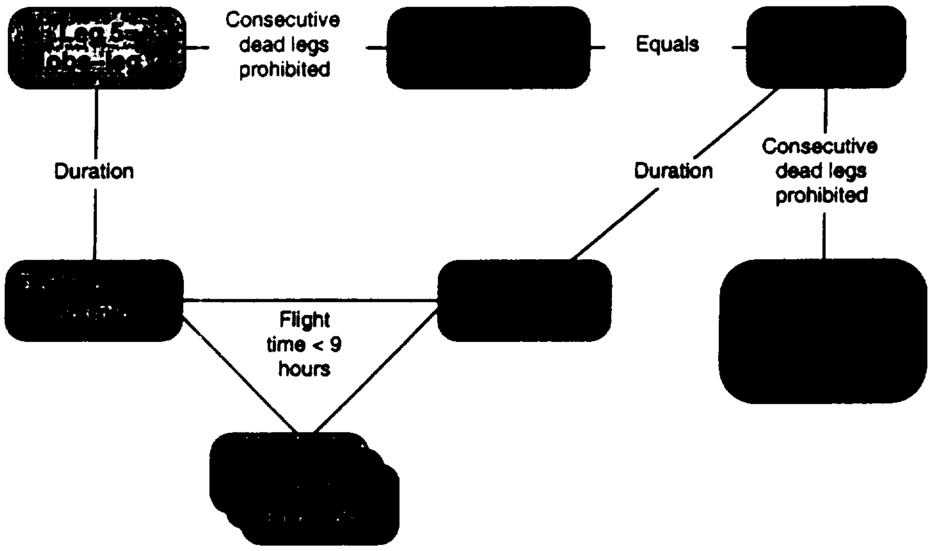
Consecutive dead legs prohibited
Equals
Duration
Duration
Consecutive dead legs prohibited
Flight time < 9 hours

Figure 3: New variables and constraints of the DCSP after adding a new flight leg.

Figure 3 shows the evolution of a portion of the DCSP when a new leg is added. The variables for leg 5 have all been assigned values which satisfy the various constraints between them. When the variable *Leg-After 5* is assigned the value *obs-leg*, this is a signal that the variables for a new leg, in this case leg 6, must be added to the problem; these variables are *Leg 6, Duration 6* and *Leg-After 6*, among others. In addition, the new constraints are inserted, among then that *Leg-after-5 = Leg-6*, and the constraint enforcing *Leg-after-6* can't be *dead-leg* or *home-leg*. Notice that the ***Flight time*** < 9 hours constraint actually must be modified, because this constraint acts on the duration of the new leg as well as the previously added legs.

Finally, we briefly discuss how procedural constraints play a role in representing this problem. Suppose that we have two high priority observations, one constrained to occur at the beginning of the flight and one at the end of the flight. After specifying these two flight legs, we are left with a situation in which the aircraft must fly in roughly the same direction in order to connect these two legs together and create a legal flight plan. An elimination procedure could check the remaining observations and eliminate those that require the aircraft to fly the wrong direction. Representing this type of implied constraint efficiently is impossible using a simple representation.

As mentioned previously, procedures can also be used to handle continuous variables by making certain they are not in the set of decision variables. For example, the end time and end location of observation legs are determined fully by the start time, start location and object being observed. Thus, there is no need to search over possible assignments to these continuous variables.

5.3 Solving CSPs

We now turn to methodologies designed to solve DCSPs. We begin with solving CSPs, then show how the basic algorithm can solve DCSPs as well.

Backtracking search constructs a solution to a CSP by selecting a variable from the remaining unassigned variables in a problem, then trying each possible value in turn. If at any point a constraint violation is detected, the procedure returns to the previous variable binding, and tries another value. If all the values of a variable are tried without success, then the procedure also returns to the previous variable and tries another value.

This conceptually simple algorithm is guaranteed to solve a CSP or demonstrate that no solution is possible. The worst-case running time for this procedure is the product of the sizes of the domains of all the variables, which is exponential in the number of variables. Consequently, backtracking is only feasible for CSPs with discrete domains. Backtracking can be easily modified to solve COPs by saving the value of the best solution found, and searching over all solutions instead of halting after finding the first solution satisfying the constraints. In essence, this is like imposing a new bound on solution quality each time the old bound is improved upon.

The performance of this algorithm depends dramatically on functions which select the next variable to choose, select the order to try values, perform fast inference to eliminate values from the domains of unbound variables, and decide which variable binding decision is responsible for a constraint violation. For instance, in the SFP, there is often a choice concerning which observation to make for a given observation leg. Since the goal is to exceed a bound on the priority, a good choice might be to select the remaining observation with the highest priority to try next. However, if this observation is too long or takes the aircraft in the wrong direction, other observations will not be possible and a poor quality flight plan will result. Consequently, modifying this choice by checking the direction the aircraft must fly may lead to better flight plans in less time. It should also be clear that using procedural constraints to eliminate bad choices for variables can save time, since the algorithm does not need to guess these values. These modifications are critical to good algorithm performance; for results on traditional scheduling problems, see [BF98] and [Nui94].

It should be clear from the discussion of the constraints in §3 that flight planners must choose from among several tradeoffs when scheduling flights. For example, an obvious tradeoff concerns whether to try a high-priority observation first, or to try an observation of lower priority which may be easier to schedule. Another tradeoff concerns when to schedule a particular observation. If the observation is constrained by a minimum water vapor threshold, for instance, then this may be satisfied by flying higher or further north [HB00]. However, these both require making observations later in the flight; care must be taken to ensure that the aircraft can still return home. Tradeoffs such as these drive the construction of both variable and value ordering heuristics, which are necessary to ensure good algorithm performance.

There are many other algorithms which can be used to solve CSPs. However, we only discuss backtracking algorithms in the interests of space considerations.

5.4 Solving DCSPs

It requires very little effort to modify the standard backtracking algorithm to solve DCSPs rather than CSPs. While standard backtracking guarantees that the set of variables and constraints is constant during the solving process, a solver for a DCSP must contend with the possibility that new variables and constraints are generated during the problem solving process. In Figure 4 we see that before checking for constraint violations, we must generate the new CSP, P' from P. For instance, if we decided to add a new flight leg, we would have to add the variables and constraints pertinent to this leg to the CSP. In order to guarantee that this process terminates, we must be assured that we only add a finite number of variables to the problem instance in the worst case.

```
procedure CompleteSearchDCSP($P$)
    generate next CSP, $P'$
    if a constraint is violated return fail
    if problem solved return success
    select an uninstantiated variable $V$
    for all values of this variable $v \in dom(V)$
      if CompleteSearch($P' \cup V := v$) == success
        return success
    end for
    replace $P'$ with $P$ if necessary
    return fail
end
```

Figure 4: Complete search.

6 Related Problems

As mentioned in the introduction, SOFIA supports many different instruments. Unfortunately, changing instruments is a lengthy job, which takes several hours to complete. Instrument changes will be minimized due to the overhead required before installation, the time required to remove an instrument and install another one, and the setup time for the new instrument on the airplane. This leads to an extended flight planning problem: rather than just planning a single flight, a series of flights with the same instrument must be planned. This problem includes constraints on which days the aircraft can fly; the aircraft will not fly on holidays, routine maintenance must be performed, and astronomers may not be available on some days. Additional no-fly constraints may be imposed by other observatory operations, such as PI flights or special events such as comet impacts or supernova explosions.

The techniques we propose in subsequent sections can be used to address the extended problem as well as the more limited SFP. However, we focus in the sequel on the SFP for simplicity's sake.

The SFP is similar to other problems important for both NASA and industry. For example, the Vehicle Routing Problem (VRP) is the problem of delivering packages to various destinations in an urban area. Typically, the problem features many trucks, with different capacities and fuel constraints, and many jobs with different time windows, ordering constraints and priorities [KPS99]. This problem does not have the complex geometric constraints that the SFP features, but shares many other similarities. Ordering constraints, package size, fuel constraints, truck capacity and distances all interact to make for a complex scheduling problem. Scheduling operations for planetary rovers [BGSW99] and inter-planetary vehicles [JMMR99] features the sequencing of science observations and their enabling activities such as sample acquisition, as well as operations to aid in navigation as well as re-charging operations. Visual spectrum science observations have constraints similar to astronomical observations; in the rover domain, for example, there must be sufficient light available. Furthermore, the total number of operations must be within the available power budget of the vehicle. Finally, path planning in the rover domain and astronomical navigation feature complex geometric constraints which may require simulation.

7 Conclusions

The flight planning problem motivated by the SOFIA GI program is a complex one, with many heterogeneous constraints over a mixture of continuous and discrete variables. The resulting problem is further complicated by the fact that the problem includes an uncertain number of steps and other conditional constraints that can be very expensive to encode in a single problem instance. These factors lead us to the conclusion that traditional scheduling techniques which solve simple scheduling problems are, by themselves, likely to be inadequate. We instead advocate representing the problem as a DCSP employing procedural constraints. This problem can then be solved by a complete search algorithm using the procedural constraints to efficiently eliminate possible assignments. Furthermore, this problem is similar to other problems which are important for NASA and industry to solve. Therefore, it is worthwhile to address these problems and gain experience in solving them; only by doing so can we ensure that SOFIA's choice will be made correctly.

8 Acknowledgments

I would like to thank Ari Jónsson, David Smith, Sean Casey, and Jesse Bregman for their help in writing this paper.

References

[Bec97] E. E. Becklin. Stratospheric observatory for infrared astronomy. *Proc. of ESA Synmposium* The Far Infrared and Submillimetre Universe, *ESA SP-401*, pages 201–206, 1997.

[BF98] J. C. Beck and M. Fox. A generic framework for constraint-directed search and scheduling. *AI Magazine*, 19(4):101–130, 1998.

[BGSW99] J. Bresina, K. Golden. D. E. Smith, and R. Washinton. Increased flexibility and robustness of mars rovers. In *Proceedings of the Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS99)*, 1999.

[Bru98] P. Brucker. *Scheduling Algorithms*. Springer-Verlag, 1998.

[ED97] E. F. Erickson and J. A Davidson. Sofia: The future of airborne astronomy. *Proc. of the Airborne Astronomy Symposium on the Galactic Ecosystem: From Gas to Stars to Dust*, 73:707–732, 1997.

[HB00] J. M. Horn and E. Becklin. Optimized flight planning for sofia. In *Proceedings of the SPIE*, volume 4014, Munich, 2000.

[J96̂] A. Jónsson. *Procedural Reasoning in Constraint Satisfaction*. PhD thesis, Stanford University, Stanford, CA, 1996.

[JF99] A. Jónsson and J. Frank. A framework for dynamic constraint reasoning using procedural constraints. In *Proceedings of the Workshop on Constraints and Control, held in conjunction with the 5th International Conference on Principles and Practices of Constraint Programming*, 1999.

[JMMR99] A. K. Jónsson, P. H. Morris, N. Muscettola, and K. Rajan. Next generation remote agent planner. In *Proceedings of the Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS99)*, 1999.

[KPS99] P.J. Kilby, P. Prosser, and P. Shaw. A comparison of traditional and constraint-based heuristic methods on vehicle routing problems with side constraints. *Journal of Constraints, to appear*, 1999.

[KS96] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the 13th National Conference on Artificial Intelligence*, 1996.

[Lue84] D. Luenberger. *Linear and Nonlinear Programming*. Addison Wesley, 1984.

[Nui94] W. Nuijten. *Time and Resource Constrained Scheduling: A Constraint Satisfaction Approach*. PhD thesis, Eindhoven University of Technology, 1994.