Dispatch Scheduling of Automated Telescopes

Robert B. Denny
DC-3 Dreams SP, Mesa, Arizona

Abstract: Automated telescope scheduling systems have traditionally focused on optimizing the use of the observatory, minimizing overhead and maximizing shutter-open time. However, most observatories do not enjoy consistently good skies. Conditions can change significantly during an observing session, leading to schedule breakage. In addition, science observing may require prompt follow-up observations that arise during a night's observing. These issues give rise to the need for a scheduling system that is capable of recovering from periods of bad skies, wind, etc., and of integrating newly added observations during operation, all without operator intervention. The concept of "just-in-time" or *dispatch scheduling*, where the scheduler dynamically makes a "best" choice for the next observation, will be discussed. A dispatch scheduler was constructed, tested initially (as described in the previous versions of this paper), deployed and supported as a commercial product then revised per user feedback and ongoing research. This experience exposed two weaknesses in the design. Solutions have been found and validated. One solution also improved the overall efficiency of the scheduler. This paper will describe the revised scheduler, the specific weaknesses encountered, and the solutions to this weakness, Rising Plan Delay and an additional efficiency function coefficient Project Completion.

Revised (5): September 2018

1 Introduction

The process of acquiring data for both astronomical science and artistic astroimaging involves *planning*, *scheduling*, and *observing*. These three phases of data acquisition may be viewed as *what*, *when*, and *how*, respectively. Managing acquired data (*where*) is a separate topic that will not be addressed in this paper. When designing tools for data acquisition, it is important to keep the three considered activities clearly separated.

1. **Planning** establishes what data is needed for the mission, and may place constraints that affect acquisition timing in order to assure that the data meets the minimum quality needed to support the mission. Typically, planning is done by the **investigator** or **astronomer**.
2. **Scheduling** makes the decision as to when requested data *can* and *should* be acquired, in order to meet the constraints. This is the role of a **scheduler**.
3. **Observing** involves the control of the observatory instruments and software to capture a package of data (which may be multiple images at multiple wavelengths, for example). The use of a scheduler implies that a **sequencer** is used to automate the data acquisition process when directed by the scheduler.

This paper, and the engineering work it describes, *focuses only on scheduling*. The implications of this may not be immediately obvious. Suppose a request is submitted for an observation with coordinates and constraints that make it impossible to observe at any time during the year regardless of weather. It is not the scheduler's job to alert the user that he has entered an impossible request. It simply will never schedule the impossible request. It is the job of the astronomer (and any planning tool) to create an observing plan that is practical as well as supportive of the mission. One can envision multiple specialized planning tools that feed their requests into the scheduler through a common protocol. As stated, these tools are not a topic of this paper.

2 Background

During the initial phase of this project, it was found that virtually all of the research on scheduling of resources had optimal resource utilization as the goal. This typically involves a complex time-consuming process, and applied to astronomical observing, produces a *static* schedule for an entire night's observing. The success of such an optimized static schedule depends on (a) problem-free execution of each observation, (b) perfect knowledge of the time duration needed for each observation, and (c) perfect fore-knowledge of the weather throughout the night.

Once an observation fails, any linked observations also must fail, leaving holes in the schedule. If the telescope needs an un-forecast refocusing, the schedule is broken, requiring observations (and possibly linked observations) to be skipped. If the sky conditions or weather changes, it can eliminate an entire class of observations from consideration due to their constraints being violated. For example, a thin cirrus layer could preclude all-sky photometry, but there could be other as-yet unscheduled observations that could be made without deleterious effect.

These considerations led to the desire to make the scheduler *dynamic* in some way, able to adapt to changing conditions, new requests, and acquisition errors, while still maintaining reasonable efficiency. Further research in this direction produced the Steele and Carter paper (*ref. 1*). The concepts discussed in their paper provided basic concepts for the design of the present scheduler. However, very little detail is contained in Steele and Carter, and the difficult problem of handling linked observations is not treated at all. The basic concepts presented in their paper will be briefly described in the following sections. No claim of originality is made for these concepts.

2.1 Scheduler Design

Steele and Carter identify the following three criteria for a "good schedule": (a) fairness, (b) efficiency, and (c) sensibility. A *fair* schedule balances time allocations between users such that they all share good and bad observing times equitably. An *efficient* schedule is one that maximizes instrument utilization and strives to match observations with required conditions. A *sensible* schedule is one that attempts only those observations that are possible under the current observing conditions.

The **ACP Scheduler** (ACPS) is a practical adaptation of Steele and Carter's concepts combined with additional features and practicalities needed to turn their basic ideas into a commercial-class scheduling engine. It is designed to handle observation requests from multiple users. Requests are kept in a permanent relational database store. The requests can be entered months ahead of the time at which constraints will first be met, or minutes before they are needed.

Throughout this paper, many of the fine-grained details and features of the scheduler are omitted for clarity. They don't affect the basic conceptual knowledge gained from user experiences and simulations.

Table 1. Scheduler Data Hierarchy

User	Top-level node. Represents a user or using organization, not necessarily an observer or investigator
Project	Child of User. Represents a scientific project that may require multiple sets of data
Plan	Child of Project. **This is the basic schedulable unit**. It represents one or more observations *that must be performed as a group and within a single night*.
Observation	Child of plan. Represents a *single target beginning at a single time*. **This is the basic unit of work for the observatory**. Constraints are applied to observations. Linked observations are entered as separate observations with the same parent plan, with specified time separations.
ImageSet	Child of Observation. Represents one or more images to be acquired *back-to-back and at a single wavelength*.

In order to understand the descriptions of scheduler behavior in later sections, it is necessary to define some terminology. The scheduler request database is hierarchical and consists of the following node types, as shown in Table 1 above.

The simplest plan is one with a single observation and a single image-set that specifies a single image. Plans with more than one observation form *linked observations* with a specified time interval between them.

Each observation carries with it a set of *constraints* that limit the times at which the observation can be taken. Example constraints include altitude, air mass, seeing, moon phase/elongation, and sky condition. The scheduler supports an open-ended set of constraints through a plug-in facility. For this paper, only a basic "above the horizon" altitude constraint was used in simulations. A more complex set of constraints would serve only to make it more difficult to interpret results.

Priorities are supported, and are applied to plans. Rather than impose a fixed priority range on everyone, each User is allowed to assign any range of priority values to *their* plans. During the scheduling cycle, (see below) priorities are normalized in a way that maximizes fairness between Users.

2.2 System Architecture

Figure 1 is a software block diagram of a robotic observatory controlled by the scheduler and a sequencer, augmented by user tools for browsing the request database and for planning observing requests.

The sequencer is responsible for automatically manipulating all of the observatory instruments and equipment needed to acquire the data for each *observation* (the basic unit of work for the observatory). The scheduler was designed with a standard sequencer interface, allowing it to potentially be used with a variety of robotic observatory control systems. Two sequencers have been developed at

this point: (1) a simulator sequencer for research and testing, and (2) a commercial observatory control system[1].

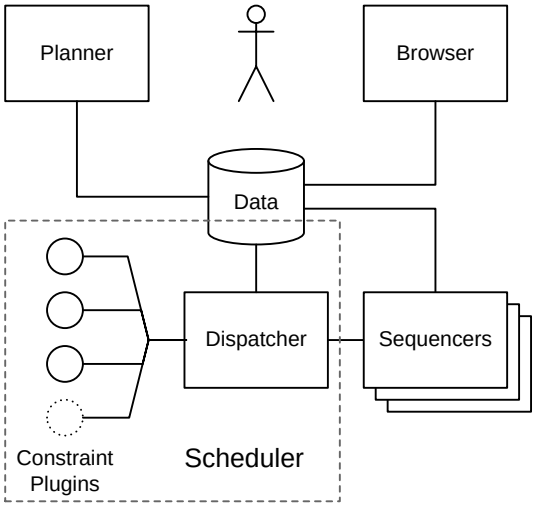

Figure 1. Software Block Diagram

The scheduler consists of a dispatcher and a set of constraint plug-ins. As described elsewhere, constraints can be separately developed, allowing for addition of custom constraints for special applications. The dispatcher is responsible for selecting plans to start, and for dispatching observations to the active sequencer.

The centerpiece of the system is a relational database which contains all of the observing requests and the current states of each. All of the components of the system communicate through the database via an operating system vendor supplied database engine plus a wrapper object that isolates the system components from low level database operations and Structured Query Language (SQL) statements. Thus, the database looks like a special purpose black box with all storage and retrieval details hidden from the system components.

3 Scheduler Operation - Overview

Fundamentally, ACPS is a *dispatch scheduler*. At each scheduling cycle, it decides which of the eligible plans to start next. Once started, a plan must run to completion or fail completely. If a plan fails, it is re-queued and will be attempted again later. It is possible (via a user control) to give preference to failed (and thus re-queued) plans with respect to those that have never been started.

When an observation is completed, the scheduler looks to see if one or more new plans *can* be started (based on constraints) before an already-running plan's next linked observation comes up for acquisition. At a minimum, the first observation of a candidate new plan must fit into the time remaining before any already-scheduled observation (belonging to any already-started plan). If a candidate new plan has multiple linked observations, *all* of them are checked against *all* of the linked observations in already-started plans. If there are any clashes, the candidate plan will not be considered. There is no requirement that linked observations be spaced at regular intervals.

3.1 Efficiency Function

After all eligible plans are checked for constraints and timing clashes with running plans, the remaining eligible plans form the set from which the scheduler picks the "best" plan to start in this cycle. As described in detail below, this is done by applying an *efficiency function* to each plan. The plan with the highest efficiency index will be started. Of course, if only one plan remains after the preceding tests, it will be started without qualification.

3.2 Concurrent plan Execution

It is important to understand that multiple plans may be active at any point in time. This will happen if any plan with more than one linked observation is active. As we will see, one of the scheduler's important tasks is to avoid starting a new plan if any of its observations would clash in time with any of the linked observations of running plans.

Any point in time, the remainder of the night may already have time slots reserved for doing linked observations of running plans. These time slots are in general irregularly spaced in time and occupy irregular intervals of time as well. Thus, multiple plans may be concurrently active, and this is an important feature of the scheduler if it is to achieve its goal of efficiency.

4 Scheduling Cycle

The scheduler runs a continuous loop consisting of the following steps (simplified for clarity):

1. Calculate estimated time spans of any newly added plans
2. Normalize priorities.
3. Start any plans which have specified an absolute start date/time, and for which that date/time has arrived. If such a plan is started, send its first observation to the Sequencer. Jump to (10).
4. Look to see if any already-started plans have an observation that is due now. If so, send that observation to the sequencer. Jump to (10).
5. Change unstarted plans that were previously vetoed and deferred by constraints, and for which the deferral time has expired, back to a status of pending (eligible to be started now).
6. Change unstarted plans that were previously vetoed due to a time clash back to a status of pending.
7. At this point, only eligible pending plans are left to consider. For *each* of them:

    A. Check to see if the entire plan can be completed before sunrise if it were to be started now. If not,

[1] ACP Observatory Control Software, developed by the author.

defer the plan, removing it from consideration until the next night.

B. Check *each* observation of the plan (if more than one observation, the second and subsequent will specify a time from the previous observation), using the current time as a baseline, as follows:

   1. Apply observer-specified constraints at the (estimated) time. If constraints cannot be met, *veto* the plan and move on to the next candidate.

   2. Test to see that the observation does not clash with any observation belonging to an already-started plan. If a clash is detected, *veto* the plan and move on to the next.

C. For a rising plan, check to see if it can safely be allowed to continue to rise even though it would now be eligible. If so, *defer* it for a bit of time.

8. If there are any eligible (unstarted) plans left at this point, pick the "best" one to start. This is the critical operation, the one that determines the behavior of the scheduler. We will expand on this below.
9. Send the first observation of the selected plan to sequencer for data acquisition.
10. Wait for completion notification from sequencer.
11. Loop back to 1.

Priority normalization is done by converting each User's plan priorities into a new value such that the mean priority of *all* of that User's plans is 0.5. This scheme came from Steele and Carter[1]. It is the fairest of all of the priority schemes studied. It is done at every pass through the scheduling loop to allow for any-time addition of new requests which may change the range of priorities for the submitting User.

Application of constraints and selecting the "best" next plan to start are the *core of the scheduling system*, and are discussed in detail below.

Step 7C above deserves additional explanation[2]. This step tries to let a rising plan continue to rise even though all of its constraints are met. Without this logic, an eligible plan will be started as soon as its constraints are met. If it is rising, this may not make the best use of the observatory, as data acquired at higher elevations is usually of better quality.

It should be noted that, after waiting for a dispatcher wake cycle, additional plans may become eligible, and the formerly single plan will have to contend with these newly eligible plans via the Efficiency function. This could be an advantage or disadvantage. Remember that, once a plan is started, all of its time slots are reserved. Suppose a more efficient plan has met constraints after the wake cycle. Shouldn't this one be run anyway? If the first plan had been started immediately, it would have blocked the second one. In general, it's better to be faced with a choice of plans and do the most efficient one.

In any case, once a plan has been chosen to start, all of its observations' time slots are reserved, and its first observation is given to the sequencer to execute. As you may recall, an observation applies to a single target and consists of one or more image-sets, each of which comprises one or more images at a single wavelength. The Sequencer performs the slew to the target, perhaps checking the pointing with a short validation and adjustment exposure, then performs an auto-focus if requested or indicated, and activates the auto-guider if applicable. It then commands the imager to acquire images per the image-sets, switching filters as needed. Each filter switch necessitates (at a minimum) a refocus, even for supposedly par-focal filters. In addition, if the image needs to be guided, and if the guiding sensor is behind the filters, the guider must be re-started with a new exposure/cycle time. Focus changes arising from a filter change are handled either by a table of focus offsets or by auto-focus (the latter is inefficient!).

5 Scheduling Rules

Early simulations led to a couple of basic rules that guided the design. Recall that the plan is the basic schedulable unit, and may consist of multiple linked observations (targets) with specified time intervals between them. These scheduling rules are:

1. Once a plan has been started, it must either run to completion (in one night) or fail completely[3]. In other words, a plan's observations must be acquired as a unit. Time spacing between linked observations, and a possible "hard start time" needed for target phasing constraints, dictate this rule.
2. Time separation of a plan's linked observations must include a non-zero time tolerance. The scheduler is not perfect; therefore the observer must indicate the allowable variation between linked observations for which the acquired data will still be usable.
3. Once a plan has been started, its linked observations must be considered inviolable. Nothing can pre-empt a running plan's scheduled linked observations.

For the simple case, a plan that has a single observation, rule 1 above is intuitively obvious. However, for a plan that has multiple linked observations, the rules mean that a plan cannot be started unless the following conditions are met:

1. All of the plan's linked observations' constraints can be met (at their scheduled time) if the plan is started now.
2. None of the plan's linked observations will clash with those of plans that have already been started, regardless of their priority.

[2] See section 7 Rising Plan Delay

[3] See section 5.1 Best-Efforts Plans wherein this rule may be optionally relaxed.

These rules lead to the following corollaries:

1. Constraints must be applied to all observations of a candidate plan before starting it. The proposed plan start time, the estimated times needed to execute each of its observations, and the specified time interval between its observations all must be used to project forward each observation in time, and compute its constraint for that time.
2. A higher priority *unstarted* plan can never force the failure of a lower priority *running* plan; it can only prevent a lower priority plan from getting started.
3. The projected times for starting each of the candidate plan's observations must be checked that they do not overlap/clash with the *nominal* start times and time spans of the remaining linked observations of plans that have already been started.
4. Once constraints and timing-clash checks have been applied to an entire plan, it can be started with a high degree of confidence that it will complete successfully. There is, however, a non-zero chance that it will fail because a constraint is not met at the *actual* time of observation. This condition can arise as a result of imperfect estimates of times needed for other observations.

It should be clear that constraints must first be applied to *all* observations of each unstarted plan, with forward time projection[4]. If any constraints are not met, the observation and its parent plan are vetoed and eliminated from consideration during *this scheduling pass*. The plan will be re-queued and reconsidered in subsequent scheduling passes, as the constraints could be met at this later time. Thus, the veto processes eliminates plans that cannot be started at the current time.

5.1 Best-Efforts Plans

Field experience with the scheduler revealed that Scheduling Rule 1 (Plans must complete in a single night) resulted in a limitation that made some types of observing impractical. For example, data acquisition for photometry applications often requires a time series which extends over a long period of time. A plan with linked observations does provide time series capability, however if the time series needs to be "long", its chances of being successfully started under Scheduling Rule 1 decrease.

The nature of a dispatch scheduler is such that the starting time of a plan cannot be predicted. This limits time series to the shortest possible length that can fit into a single night when the plan is started at the latest possible time. This limitation led to a requirement for modifying Scheduling Rule 1 to allow a plan to be started *without* checking to see that it could be completed before being stopped by constraints or dawn. In addition, while the plan is running, a veto or failure of a linked observation in the time series causes the Plan to be successfully completed, not failed.

This mode of treating plans could be called "do as much as you can" or "best-efforts". The revised scheduler was modified to include an optional best-efforts flag on plans. If the plan is flagged as best-efforts then (a) it will be eligible to be started based only on its first observation's constraints being met, and (b) while running, if a linked observation is vetoed or fails, the Plan is marked completed instead of failed.

6 Application of Constraints

The scheduler has a standard set of constraints (see Table 2) that *may* be applied. If no constraints are applied, plans will be eligible based solely on their time span, astronomical night (Sun below –18 degrees), and instrument limitations retrieved from the Sequencer.

Besides testing constraints themselves, the constraint plug-ins also calculate a time estimate for which the "allow" or "veto" condition will remain. This is very important for scheduling efficiency, as the latter allows plans to be skipped during dispatching until the expiry time for the veto state, and the former provides time limit guidance for step 7-C in **Section 4 Scheduling Cycle**.

Table 2. Standard Constraints

Horizon	Observation must be made above the given elevation (with respect to the local mathematical horizon)
Air Mass	Observation must be made at or below the given air mass.
Sky Quality	Observation must be made at given (or better) sky quality. Four sky qualities are defined: excellent, good, fair, and poor.
Dark Time	Observation must be made with Moon "down", namely 2 degrees below the mathematical horizon
Moon Distance	Observation must be made when the target and the Moon are separated by at least the given angular distance
Moon Avoidance	Observation must be made at or below the given moonlight level. This is expressed in terms of a Lorentzian weighting that is a combination of angular distance from the Moon and the illumination at its current phase.

6.1 Strict versus Lenient Application

The first version of the scheduler applied constraints only for the starting time of each observation, in order to minimize scheduling overhead. This was found to be insufficient for practical use. With this *lenient* application of constraints, it is possible for an observation to fall out of constraints during acquisition. Thus, at least some of the acquired data could fail to meet the requirements of the constraints.

[4] See section 6 Application of Constraints for details.

In the revised scheduler, application of constraints was changed to a *strict* model, wherein constraints are checked for both the starting time and the estimated ending time of each observation. Only if constraints are met at both of those points in time is the observation considered eligible for execution.

6.2 Custom Constraints

In addition, custom constraints can be added via a plug-in API. Thus, applications with unusual requirements can be supported without changes to the main dispatcher engine. Custom constraints may be developed apart from the code base for the scheduler, and simply dropped into a specific directory. The next time the scheduler is started, the new constraint plug-in is detected, additional tables and relations are created in the schedule database, and the new constraint will appear in the schedule browser's user interface.

7 Rising Plan Delay

Real world experience and parallel simulations indicated a significant weakness when the scheduler is under-subscribed. Suppose that, during a particular dispatch cycle, there is only one eligible plan left after application of constraints. In this case the Efficiency function is not applicable. There is but one "best" choice. That single plan is dispatched immediately. As the run progresses through the night, this increasingly happens very shortly after the plan first meets its constraints, on the rise.

As a result, the plan will be started well east of the optimum sky position, and the data will be acquired through nearly the *maximum* air mass allowed by its constraints. Figure 2 shows the typical behavior of a moderately loaded dispatch scheduler without rising plan delay.

As the scheduler's loading increases, this effect lessens because it's more likely that multiple plans will be eligible in a dispatch cycle, at least one will be higher than it needs to be, and the Efficiency function will be able to pick the best one. The problem is absent in a heavily loaded situation as there are always plans near the meridian and the Efficiency function does its job well.

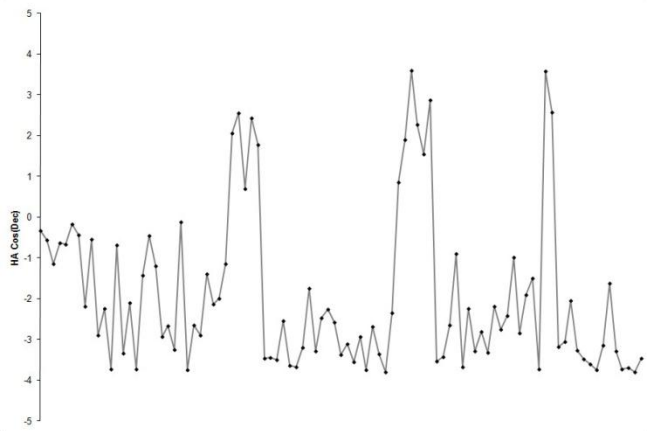

Figure 2. Meridian position with typical dispatcher

In principle, the solution to this problem involves letting rising plans[5] continue to rise even though they are eligible. But how long should they be allowed to rise beyond the point at which they can first be started? Clearly, a rising plan cannot (or should not) be allowed to rise beyond any of the following points:

1. The time at which it no longer meets its constraints
2. The time beyond which it would extend past dawn
3. The time at which the Plan's centroid gets within half of its time-span of the meridian. If started at this point, the images will be acquired as close to the meridian (on either side) as practical.

But there are other less obvious considerations. Suppose one or more additional plans become eligible in the future? It is possible that one of these newly eligible plans would be selected by the Efficiency function in preference to the delayed rising plan. If the selected plan's first observation exceeds the allowable times 1-3 above, the delayed rising plan will fall out of constraints before this new plan's first observation completes. The delayed rising plan will never be started.

It should be clear that the more heavily loaded the scheduler is, the more likely the above scenario will arise. Thus, in general, delaying a rising plan's start while it rises reduces its chance of being run. The longer it is delayed, and thus the closer it gets to one of the limits 1-3 above, the more likely it will not be run, or run past the meridian.

On the other hand, if this policy is applied evenly, where *all* rising plans are delayed per the same policy, the effect will be to shift the start times of all such plans toward the future, improving their observing conditions. Furthermore, at the start of the run there will be both rising and setting plans. By delaying the rising plans, the dispatcher will start with the setting plans, which is desirable since they generally have shorter eligible lifetimes.

Finally, as the schedule becomes more loaded, there will more likely be eligible plans in a more favorable position, and these will be selected by the efficiency function (see below) anyway. By allowing the less well positioned plans to rise, the effect is the same as simply keeping them eligible and letting the scheduler pick the most favorable. So as the schedule becomes more heavily loaded, the less will be the effect of delaying rising plans. This was proven in the simulations.

The decision to delay a rising plan must take into account the shortest of limits 1-3 above, and the potential benefit of allowing it to rise further. Furthermore, the amount of time to defer the rising plan (once it has been decided to defer it in the first place) should be shorter than the threshold of the time-left used to make the deferral decision. This allows for some tolerance in time estimates. Algorithm complexity could quickly reach a point of diminishing returns while impacting scheduler performance.

[5] The "coordinates" of an entire plan are given by its *centroid* as defined in section 8.2 below.

Therefore, it was decided to implement a simple algorithm for rising plan delay: If the plan's centroid is rising, and if there is more than 10% of its time span remaining in the shortest of limits 1-3 above, defer the plan. If neither of the "hard" limits (1 or 2) will be reached, this means the Plan will be started when its centroid comes wiothin half of its time-span of the meridian, plus another 10% lead time.

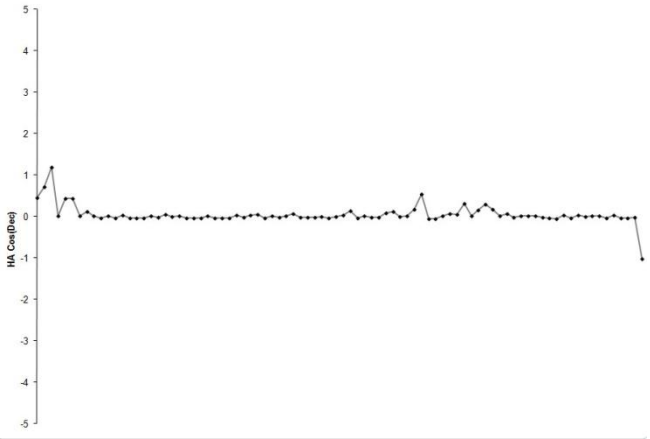

Figure 3. Meridian position with rising plan delay

Figure 3 shows the behavior for the same set of requests as Figure 2, but with the rising plan delay algorithm included. The beneficial effects are clear. Note particularly that more images were acquired because the westernmost plans were immediately started while the more eastern ones were deferred for rising plan delay[6]. Extensive simulations of this algorithm have shown that delaying rising plans is surprisingly effective when loading is light, and has an unexpectedly low impact on efficiency and total number of targets acquired when loading is heavy.

8 Efficiency Function

After application of constraints and rising plan delays, each remaining eligible plans is tested by computing the *efficiency index*. The eligible plan with the highest efficiency index is the one chosen to start now. The purpose of the efficiency index calculation is to decide which plan to start considering both the scientific priority and the best use of the current observing conditions. This is done using an Efficiency function of the form:

$$\mathbf{E}(n) = \sum_{k=1}^{8} \beta_k E_k(n)$$

where *n* indexes the plan under consideration, and *k* indexes the eight efficiency terms described below.

This generic form is taken from Steele and Carter[1]. However, the specific terms of this function, both the semantics of some the efficiency terms $E_i(n)$, as well as their coefficient values $ß_i$, differ in from Steele and Carter. In addition, new efficiency terms were added after experience gave rise to their need.

The knowledge gained via the simulations and user feedback provided insight into the behavior of a dispatch scheduler, and led to two sets of standard terms and $ß_i$ coefficients (weights) suitable for most observing tasks. These are described following the descriptions of the individual terms, in section blah.

The scheduler also has a mode in which the user can adjust the weights, giving complete flexibility. There is no plug-in interface for adding efficiency terms; but individual terms may be disabled by setting their weight to zero. By varying the weights, the behavior of the scheduler can be adjusted to meet virtually any need, or to conduct engineering studies using the simulators.

After research, the details of which are beyond the scope of this paper, the following efficiency terms/functions were chosen for the scheduler:

8.1 Scientific Priority

The scheduler allows each User to assign their own scientific priorities to their plans rather than forcing everyone onto a single priority system. In order to assure that allocation of the observatory is fair, each User's priorities are transformed into a *normalized* system where the mean value of their priorities is 0.5:

$$\frac{\sum_{i=1}^{N} p_i}{N} = 0.5$$

where *N* is the total number of plans in the system for that User. The normalized priority for each plan *p* is stored in the scheduler's database and used in the Efficiency function calculation as shown below.

It is planned for the future to study whether the priority of a plan should be scaled according to its number of linked observations. This would assign a weight proportional to the resources that the plan uses. A further refinement might be to weight according to the observatory time needed for the plan.

In any case, the candidate plan's normalized priority *p* is used to calculate the E_1 term of the Efficiency function, thus

$$E_1(n) = p(n)$$

A front panel control "Ignore Priority" is provided so that the scheduler user can toggle between $ß_1$=0 and the standard $ß_1$ value. This is useful in special situations where the user wants to eliminate preference based on scientific priority. It is ignored if the scheduler's efficiency mode is set to Custom. The control is ignored if the scheduler's efficiency mode is set to Custom.

8.2 Nearness to Transit Altitude

It is intuitively obvious that it is advantageous to observe some objects at as low an air mass as possible. The

[6] This was confirmed by inspecting the coverage charts; the run in Figure 2 failed to get the westernmost targets.

simple interpretation of this would imply an *E* function of the form

$$E_2(n) = \frac{A_C(n)}{90°}$$

where $A_C(n)$ is the current altitude of the first observation of the candidate plan *n*. However, this would unfairly favor objects whose declination is near the latitude of the observatory (as observed by Steele and Carter[1]). A better criterion is the distance of the object from its transit altitude. This implies an *E* function of the form

$$E_2(n) = \frac{A_C(n)}{A_T(n)}$$

where $A_C(n)$ and $A_T(n)$ are the current and transit altitudes of the first observation of plan *n*.

But there is an additional consideration that is non-intuitive but became obvious after early simulations: If the candidate plan contains multiple linked observations of *different* targets, it would be incorrect to use the current altitude of the first (or any other) of the plan's observations in the above test.

Instead, the scheduler uses the *centroid* of all of the plan's linked observation equatorial coordinates as the "coordinates" of the plan as a whole. The centroid coordinates are then converted to altitude using the local sidereal time (LST) projected forward from the current LST by half of the plan's calculated time span, yielding a centroidal altitude $\bar{\mathbf{A}}$ for the plan as a whole. Thus,

$$E_2(n) = \frac{\overline{\mathbf{A}}_C(n)}{\overline{\mathbf{A}}_T(n)}$$

where $\bar{\mathbf{A}}_C(n)$ and $\bar{\mathbf{A}}_T(n)$ are the current and transit centroid altitudes of plan *n*.

Simulations revealed that the centroid method is an excellent way to treat a plan with multiple linked observations at possibly different coordinates. It turns out that the plan is most often started at an efficient time, and the individual observations are done as closely as practical to their transit altitude, on average.

In section 8.7 below, an experimental *alternative* to Transit Altitude (called Highest Altitude) is discussed. If its weight E_7 is greater than zero, then E_1 must be set to zero as these two terms are mutually exclusive.

8.3 Slewing Overhead

It is more efficient to observe nearby targets when possible, so a slewing overhead term is included in the Efficiency function. Considering only the time needed to slew to the target unfairly penalizes observations that take a comparatively long time to complete. For example, if a candidate observation is expected to take an hour to complete, a thirty-second slew is not significant. If the observation consists of a single ten-second exposure, the thirty-second slew has a significant impact on efficiency. Thus, slewing overhead is represented by an *E* function of the form

$$E_3(n) = \frac{t_O}{(t_S + t_O)}$$

where t_O is the estimated time needed to complete data acquisition for the observation, and t_S is the estimated slewing time needed to get to the target coordinates of the *first* observation of the candidate plan before starting data acquisition. The sequencer provides the scheduler with a time-to slew estimate, given a new target's position, and this is t_S above.

Note that this term does not consider the time needed to slew to possible subsequent linked observations in the candidate plan. There is no way to know the starting point for such slews, as the scheduler is intrinsically dynamic.

A front panel control "Prefer Short Slews" is provided so that the scheduler user can toggle between $ß_3$=0 and the standard $ß_3$ value. The control is ignored if the scheduler's efficiency mode is set to Custom.

8.4 Retry Count

The scheduling rules state that a plan must either complete successfully within a night, or fail completely. In cases where a plan fails due to changes in sky condition, weather shutdowns, or (rare) scheduling errors that cause an observation's specified time window to be missed, the scheduler re-queues the plan, making it eligible to be started again. This could be in the same night (if constraints can still be met) or it may cause it to be delayed until a succeeding night.

In order to provide some level of preference to failed and re-queued plans, the scheduler keeps a count of the number of times a plan has been re-queued due to failure. This retry counter is used to provide a boost in preference in the Efficiency function. This term is represented simply as

$$E_4(n) = R\,(R \le 3)$$

where *R* is the retry count (= 0 if the plan is being started for the first time). Furthermore, *R* is limited to 3, preventing a repeatedly failing plan from being unfairly weighted.

The intention is to have the $ß_4$-weighting coefficient set to a low value, providing only a mild boost in priority for re-tried plans. Setting this to a high value could cause a failed plan to become stuck in a failure loop. Further study is planned to look for instabilities that might be caused by this term in the Efficiency function.

A front panel control "Prefer Failed plans" is provided so that the scheduler user can toggle between $ß_4$=0 and the standard $ß_4$ value. The control is ignored if the scheduler's efficiency mode is set to Custom.

8.5 Meridian Crossing

When the telescope is on a German equatorial mount, a high cost is associated with every crossing of the celestial

meridian. The mount must "flip", which can take considerable time. Besides the actual flip time, additional time may be needed to assure precise pointing to the sky after the flip due to non-orthogonality between the right ascension and declination mechanical axes.

Thus, meridian crossings have a significant impact on efficiency. The scheduler includes a meridian crossing "penalty" term in the Efficiency function, as

$$E_5(n) = 1 - M$$

where M is 1 if a meridian crossing is required to reach the observation's target coordinates, and 0 if no meridian crossing is required.

For non-German mounts, the $ß_5$ weighting coefficient is set to 0, effectively eliminating this term from the Efficiency function.

A front panel control "Avoid GEM Flip" is provided so that the scheduler user can toggle between $ß_5$=0 and the standard $ß_5$ value. The control is ignored if the scheduler's efficiency mode is set to Custom.

8.6 Lateness

Early in the development and testing of the scheduler, it became apparent that, with a moderate to full load, targets which are setting in the west during the evening might be left behind as the scheduler concentrates around the meridian. Since a given target sets even earlier on subsequent nights, the problem worsens until the target becomes completely unreachable for many months.

The Rising Plan Delay algorithm can help this somewhat as previously described. Nonetheless, it was determined that a "lateness" term was needed for applications where westernmost targets are more important then those at the meridian. The scheduler has a mode selector that allows the user to select "Prefer Meridian" versus "Prefer West". The selector enables either the Transit Altitude or Lateness terms, respectively. The two terms are never used together in the standard modes of scheduler operation.

Ideally, when calculating a lateness term, the time remaining before the (currently eligible) plan again becomes ineligible for any reason should be used. Thus

$$E_6(n) = 1 - (\Delta t_{rem}(n)/6)$$
$$for\ E_6(n) < 0\ E_6(n) = 0$$

where $\Delta t_{rem}(n)$ is the time (hours) until the candidate plan would become ineligible due to falling out of constraints or falling below the observing horizon.

8.7 Highest Altitude

Field experience and simulation revealed a limitation in the Transit Altitude term. Some rising targets will never reach their transit altitude before dawn. This led to an experimental *alternative* to the Transit Altitude term.

This new term uses the highest altitude reached by a *rising* plan's centroid during the current observing night. For targets which have already transited at dusk, the transit altitude is still used, avoiding interaction with the Lateness term described in section 8.6. For targets that do reach transit altitude during the night, its effect is identical to Transit Altitude. However for eastern targets which rise late and don't transit, the Highest Altitude term will give the same boost level for those targets' highest altitude as does Transit Altitude for setting targets and targets that *do* transit during the night.

The term is calculated using the candidate plan's centroid, as in Transit Altitude, but instead, for rising targets only, using the highest altitude reached by the target before dawn. Thus

$$E_7(n) = \frac{\overline{\mathbf{A}}_C(n)}{\overline{\mathbf{A}}_H(n)} \text{ (rising)}$$

$$E_7(n) = \frac{\overline{\mathbf{A}}_C(n)}{\overline{\mathbf{A}}_T(n)} \text{ (other)}$$

Where $\mathbf{A}_H(n)$ is the highest altitude reached by the nth plan's centroid before dawn, for a rising target, and $\mathbf{A}_T(n)$ is the transit altitude as before. If E_7 is set to a non-zero value, then E_1 *must* be set to zero, and vice versa.

The effect of this term is still under investigation. Early results are encouraging, and if eastern targets' efficiency is improved without significant effect on transiting targets, it will be adopted *in place of* Transit Altitude. At present, the scheduler supports both terms, and E_7 may be activated in place of E_1 by adjusting the weights in custom efficiency mode.

8.8 Observing Conditions

One of the scheduler's standard constraints is sky condition. Application of this constraint prevents a plan from getting started if sky conditions are poorer than required. However, if sky conditions are *better* than required, efficiency dictates that the better conditions should not be wasted. If there is a lower priority plan whose first observation *requires* the better conditions, it should perhaps be run in preference.

The simplest scheme would be to require that observations be made only at their required conditions. This is not efficient, though, as it would prevent usage in better conditions than needed even when there is nothing else to do. Instead, we use a term suggested by Steele and Carter.

In the scheduler, sky condition can be one of four values, excellent, good, fair and poor. We assign numeric values of 3, 2, 1, and 0 to these conditions, respectively. Then we calculate the E term as

$$E_6(n) = \frac{1}{|(C_R - C_A)| + 1}$$

where C_R is the required condition number and C_A is the actual condition number.

8.9 Standard Efficiency Modes

It should be clear that end users will be bewildered by the effects of adjusting efficiency weights and the resulting changes in behavior of the scheduler. Therefore, the scheduler was designed with user controls allowing two standard modes of operation, plus a third mode in which the efficiency weights are user-adjustable:

1. Prefer Meridian
2. Prefer West
3. Custom

In addition, as previously described, the Priority, Slew Distance, Meridian Crossing, and Retry Count terms may be enabled or disabled via user controls. When enabled, and when the scheduler is running in mode (1) or (2) above, standard weights are used. When disabled, the corresponding weight is set to zero, eliminating the term from the efficiency calculation. To summarize, these additional user controls are:

1. Prefer Short Slews (slewing overhead)
2. Avoid GEM Flip (meridian crossing)
3. Prefer Failed plans (retry count)
4. Ignore Priority (priority)

Note that the Ignore Priority control is provided for engineering purposes; ordinarily the user would never suppress the effect of scientific priority. Table 3 shows the efficiency weights for the two standard scheduling modes:

Table 3. Standard Efficiency Weights

Term	*Prefer Meridian*	*Prefer West*
Priority	1.0	1.0
Transit Altitude	0.7	0.0
Slewing Overhead	0.4	0.4
Retry Count	0.5	0.5
Meridian Crossing	0.5	0.5
Lateness	0.0	0.7
Highest Altitude*	0.0	0.0
Observing Conditions	0.4	0.4

* Experimental, available only in Custom mode.

9 TO Interrupt Facility

Another issue that appeared during usage of the first version of the scheduler was the need for some sort of "target of opportunity" (TO) interrupt. The dispatch scheduler is well-suited to this requirement. The obvious use case is Gamma Ray Burst (GRB) follow up. The transient nature of GRBs is such that follow up observations must begin within a few minutes of a detection by one of the satellites such as Swift[7].

Images being acquired during scheduler operation may span many minutes. Thus, it was determined that the scheduler must have a way to immediately halt data acquisition by the sequencer and optionally stop any running plans (those which have uncompleted linked observations). The latter is needed in order to make way for newly added urgent observing requests for GRB follow up. The fact that the scheduler can accept new observing requests while running makes this sort of thing possible.

To make this facility most general, it was decided to provide an externally accessible application programming interface (API) for monitoring tools. This API not only provides the interrupt signaling capability, but also a set of functions that monitoring tools can use to determine the observability of a potential TO. Uses of this facility will be the subject of a future paper.

10 Simulation Design Issues

The initial development of the scheduler used simulations throughout. In the early design phase, narrow-focus simulations were used to evaluate various candidate terms in the Efficiency function. These simulations are beyond the scope of this paper. They served primarily to assist in the selection of terms in the Efficiency function.

Once the framework was integrated into a working scheduler, a second phase of simulations was undertaken to investigate its behavior, look for anomalies, and get some feel for its performance under various conditions. In particular, the effects on behavior due to variations in the $ß_i$ weighting coefficients were studied.

After the initial release of the scheduler to commercial users, their feedback was used to drive further refinement of the design and implementation. This ongoing user-driven dev elopement yielded, among many other things, the Rising Plan Delay algorithm described in Section 7. It should be noted that, in contrast to most other observatory schedulers, the present scheduler was designed to withstand the rigors of widespread usage by non-technical astronomer/users and the wide variations in requirements of these users.

The May 2006 revision (3) to this paper occurred simultaneous with a review of the scheduler, following 2 years of deployment in the field at over a dozen sites. Part of this review included a new round of simulations following design changes. The primary purpose of these simulations was to support regression testing (assurance that changes did not impair performance and/or reliability). It was found that the design changes during the review (including the addition of the Rising Plan Delay feature) were in fact improvements in most cases, and that there were no regressions.

The January 2018 revision (4) of this paper adds a new feature, Plan Completion, to the dispatch decision process. To support development and testing, the simulator was enhanced to allow generation of realistic requests whose structure accurately reflects the typical use by astro-imagers. A new simulation section was added for this special application and the new Plan Completion feature.

[7] See http://swift.gsfc.nasa.gov/docs/swift/swiftsc.html

These simulations will be described along with a few illustrative results.

Table 4. Load Generator Mission Types

Mission type	Fraction of total load*	Description
Random single image	0.6	Single exposure, random interval 240 sec mean, 60 sec. std. dev. Random priority mean of 5, std. dev. of 2.
LRGB Astrophotography	0.2	Per-target Projects, each with multiple Plans of 30 minutes length, with varying exposures in each of 5 filters (LRGBHa) exposure of 300-600 sec. Scientific priority of 3.
Asteroid/Comet search and follow-up	0.2	4 observations, each image 180 sec. integration, spaced 45 min apart with a +/- 10 min tolerance. Scientific priority of 5.

* with all workload types turned on.

10.1 Input to Simulations

In order to provide real-world conditions for simulations, a built-in facility is provided for generating a Projects consisting of multiple plans of various kinds. The generator is capable of creating plans that are representative of the astronomy missions shown in Table 4. When generating a test workload, it is possible to selectively include or exclude each of these mission types.

The fraction of the total workload represented by each mission type is variable. In Table 4, the "fraction of total load" values given are those that result if all of the mission types are selected. If one or more mission types are disabled, the relative mix changes based on the relative frequency of the remaining mission types. Some simulations used a sub-set of the remaining plan types, and this will be clearly indicated in the description of the simulation.

Target/observation locations are generated randomly above 35 degrees elevation over the "dark sky" for the entire night on the date and geodetic location set in the scheduler. For plans with linked observations of different targets, it is possible for targets to be unreachable due to the timing. This is a real world.

Finally, the workload for the night can be set to one of the following levels:

1. Lightly booked (20% of the night)
2. Fully booked (70% of the night)
3. Over-booked (150% of the night)

The percentages refer to the amount of time that all of the scheduled observations are estimated to require, not just shutter-open time. The overhead times are taken into consideration.

10.2 Multi-Night Simulations

In order to support long-term studies (such as the effect of Plan Completion), the simulator can be set to run continuously night after night. At the end of each simulated night, the engine log and run detail logs are rotated and closed, then the time jumps to the next night's opening time. This process will continue until the dispatcher is manually stopped.

When doing multi-night simulations, the workload levels described in the previous section don't have the same meaning. The workload is multiplied by a "number of nights" input, so in reality the first night will be much more heavily loaded, etc.

10.3 Simulated Sequencer

In order to create as realistic an environment as possible, a simulated sequencer was built and attached to the scheduler's' sequencer interface. The simulated sequencer looks at the dispatched observation and its image-sets, and simply creates a time delay equal to that which a real observatory would require to complete the observation. The following common process items are given separate time estimates. The actual values of the timing parameters are shown in Table 5 below.

1. **Slewing time** (based on rates and settling time, runs start with telescope at parked position 0 HA, 0 Dec)
2. **Guider startup time** (for "long" images only)
3. **Filter switching time** (assumes focus offsets supported)
4. **Imager download time** (varies by binning)
5. **Post processing time** (plate solving, calibration, stacking)

The sequencer simulator can be configured to add a random variation to the timing values. This is used to test the robustness of the dispatcher in the face of inaccurate time estimates. In addition, the sequencer simulator can be configured to fail observations randomly. Failure of any of the images in an observation will cause the observation (and the plan) to fail, so the more images are in a plan, the more likely it is to fail, all else being equal.

Table 5. Sequencer Simulator Timings

Slew rate	3.5 degrees/second
Slew settling	10 seconds
Guider startup	30 seconds
Minimum unguided exposure interval	120 seconds
Imager download	20 seconds
Filter switching	20 seconds
German flip	90 seconds
Auto-focus	60 seconds
Image post-processing	5 seconds
Timing Noise (uniform distribution)	5% of interval*

* Disabled for some simulations.

10.4 Time Simulation

It is clearly required that simulated time be accelerated for scheduler simulations. Since scheduling itself is a CPU and disk/database bound activity, it is not clear how to treat scheduling time as part of the overall observatory efficiency. The solution is to accelerate the clock *only* during the time the sequencer simulates acquiring the data for the dispatched observation and image-sets and when the scheduler is sleeping (no work to do). The clock runs at real time during the scheduling phase. This most accurately reflects the effect of scheduling time on overall efficiency. All sources of time, including time stamps in the log file, come from the 2-mode clock.

11 Simulations and Results

This section presents the results of some of the simulations, showing the effects of varying the $ß_i$-weighting coefficients on timing and hour angle at acquisition. Two data sets were generated: one consisting of random targets of one exposure each, with random exposure intervals and random priorities, and the other consisting of a mixture of the random targets and sets of random targets of four time-spaced linked observations of each with fixed exposure intervals and priorities. See Table 4 for specifics.

The number of targets was chosen for a moderate (70%) load and an overloaded (150%) level. The test loads were generated by using the timing information in Table 5, a latitude of 33N, longitude of 111W, a starting time at astronomical twilight on March 25, 2006 (UTC), and generating random targets computing the total time needed to acquire each image of that target, and adding that to a running total. Since slewing time is not known (it is order-dependent) a guess of 45 degrees is used. When the running total reached 70% of the total night-time from astronomical twilight to astronomical twilight, generation was stopped. For example, since the random targets also had randomly varying exposure intervals, sometimes the guider would be needed (incurring additional guider startup time).

Each of these target sets were simulated over a night three times, with each of just three of the terms in the Efficiency equation set for $ß_I = 1.0$ and the rest set to 0.0 (disabling them). The three terms studied were Priority, Transit Altitude, and Slew Time.

11.1 Random Single Images – Moderate (70%) Load

The first set of simulations uses a Project consisting of 69 plans, 1 observation and 1 image-set in each with no constraints apart from being above the observing horizon of 25 degrees. Figure 4 shows the distribution of targets in equatorial coordinates.

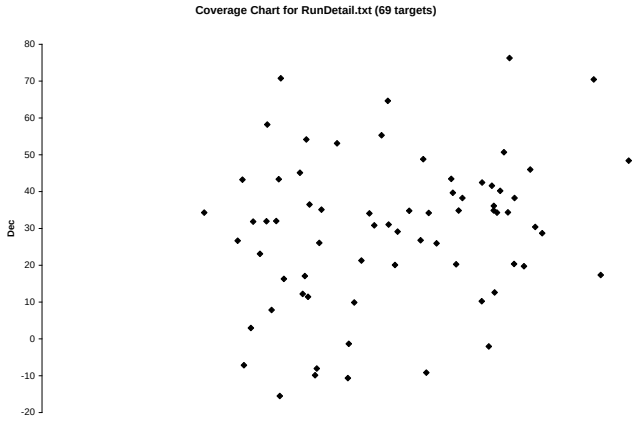

Figure 4. Targets for single image moderate load test

Priorities are random with normal distribution, mean of 5.0, and standard deviation of 2.0. Exposure intervals are random with normal distribution, 240 second mean, standard deviation of 60 seconds.

11.1.1 Priority Only (moderate load)

The first simulation with the random-images data set was run with only the Priority term in the Efficiency equation. This caused the dispatcher to always pick the eligible plan that has the highest scientific priority. Figure 5 below shows the resulting distribution of acquisition locations relative to the meridian.

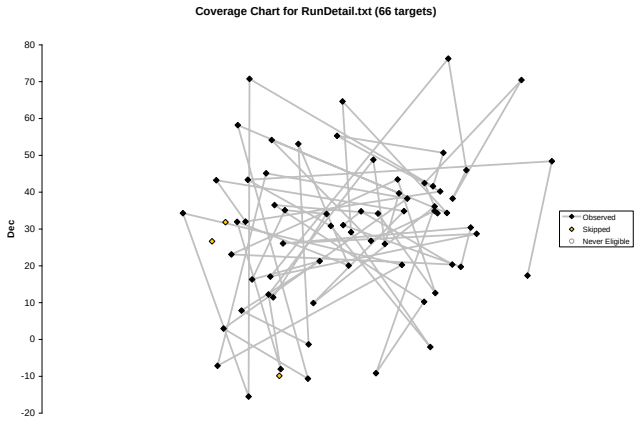

Figure 5. Target sequence for 100% priority (moderate)

For this test, 66 of the 69 available plans were completed. Analysis of the log file of the run confirmed that the plans that were not run were those with the lowest scientific

priority. Because, in this scenario, the targets were picked without regard to position and slewing time, the excessive motion (clearly visible in Figure 5 and Figure 6) adversely affected the efficiency of the dispatcher.

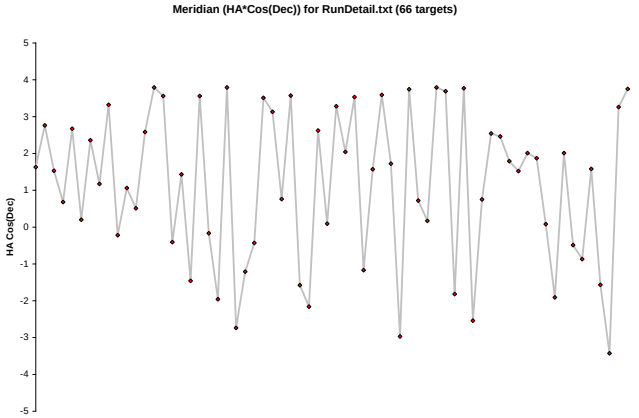

Figure 6. Meridian chart for 100% priority (moderate)

11.1.2 Transit Altitude Only (moderate load)

Next, the same random-images data set was re-run, this time with only the Transit Altitude term in the Efficiency equation. This caused the dispatcher to always pick the eligible target that is closest to its transit altitude. The results of this test are shown in Figure 7 and Figure 8 below. Predictably, most motion was in declination and the deviation from the meridian was far less. As the scheduler ran out of plans to choose from, it started picking the remaining targets which were all to the east of the meridian, so the trend is to the east (positive hour angle).

In this case, 65 of the 69 available plans were completed. Analysis of the log file revealed that those that were missed were early evening targets (lowest right ascension) whose meridian passage had already occurred. At the beginning of the night, there are plenty of targets approaching their meridian passage, and these are of course given preference in this scenario (transit-altitude-only). By the time the dispatcher got low on eligible targets, those early targets had dropped below the 25-degree altitude limit in the west.

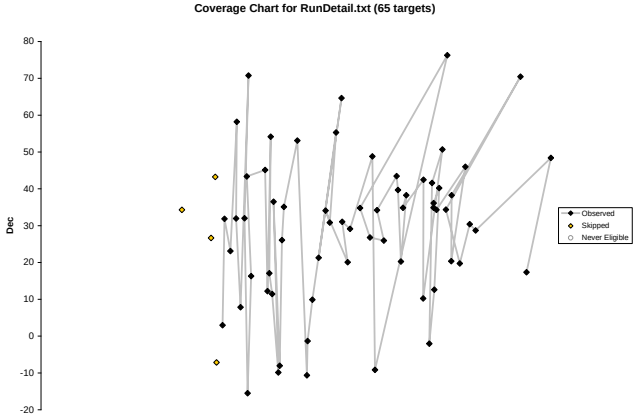

Figure 7. Target sequence for 100% transit alt. (moderate)

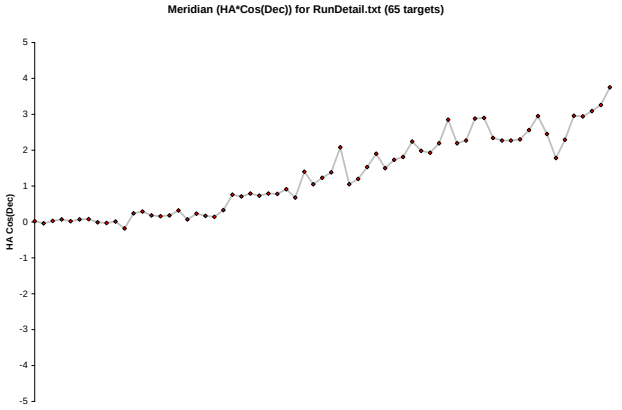

Figure 8. Meridian chart for 100% transit alt. (moderate)

11.1.3 Lateness Only

Next, the effect of the (revised) Lateness term was investigated at moderate load. This resulted in the dispatcher trying to choose the most westward targets, as shown in Figure 9 and Figure 10 below. Again, however, when the dispatcher ran out of targets, it started picking the ones to the east as they rose and came into constraints. In this test, all 69 possible plans were successfully completed, since the dispatcher started with those that were about to set, rather than starting near the meridian while some westward targets set and became inaccessible, as in the previous test.

Figure 9. Target sequence for 100% Lateness (moderate)

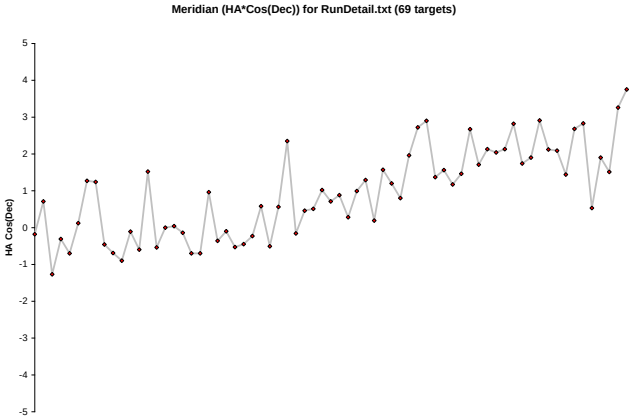

Figure 10. Meridian chart for 100% Lateness (moderate)

11.1.4 Slewing Distance Only (moderate load)

The random-images data set was run again with only the Slew-Overhead term in the Efficiency function. This caused the dispatcher to always pick the plan whose target is closest to the previous one. The results of this test are shown in Figure 11 and Figure 12 below. Minimizing slew time allowed the all 69 plans to complete, and as both charts show, the dispatcher usually selects the closest target from the previous one, wandering across the sky to achieve this goal. Slewing distances become large towards the end of the run as it does the final few targets.

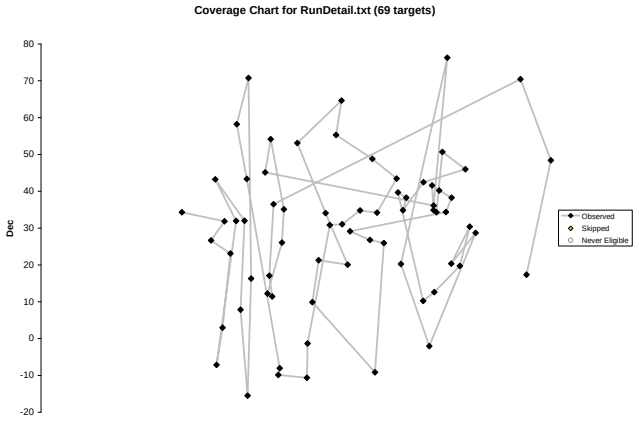

Figure 11. Target sequence for 100% slew dist. (moderate)

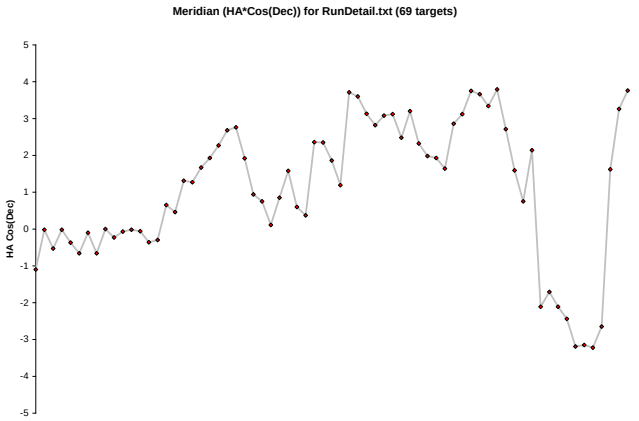

Figure 12. Meridian chart for 100% slew dist. (moderate)

11.2 Random Single Images – Overload (200%)

The effects of the priority and slewing distance terms in the Efficiency function do not change with loading. Therefore the over-loaded tests are presented only for the second and third cases above (Transit Altitude and Lateness only).

In this series, the number of requested targets was increased to 200% of the estimated maximum targets that could be acquired (again given the same observing overhead times as well as imaging times as in the previous tests). This resulted in 188 total target requests. The individual targets were generated using the same random priorities and exposure intervals as already described (though priority was ignored in these tests for clarity, as already explained).

11.2.1 Transit Altitude Only (over-loaded)

The target sequence chart in Figure 13 shows the sky distribution of the targets used in the 200% over-loaded tests, as well as the targets actually acquired when using only the transit altitude term in the Efficiency function. As expected, both early and late targets were skipped, in preference for those that happened to be nearest to the meridian at each dispatch cycle. This resulted in targets that were neither early nor late also being skipped, simply because there were too many to acquire.

The meridian chart in Figure 14 clearly shows the effect of the transit altitude term of the Efficiency function when the scheduler has more work requested than it can do. At all times there are many plans eligible, thus the Efficiency function directs the scheduler to choose the plan nearest to the meridian as its next best. Only 93 plans were successfully completed, however.

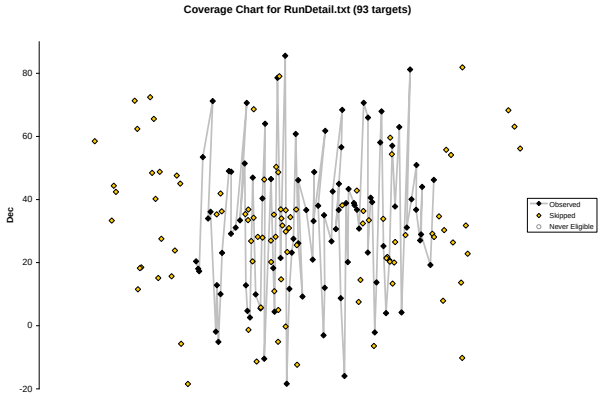

Figure 13. Target sequence for 100% transit alt. (overload)

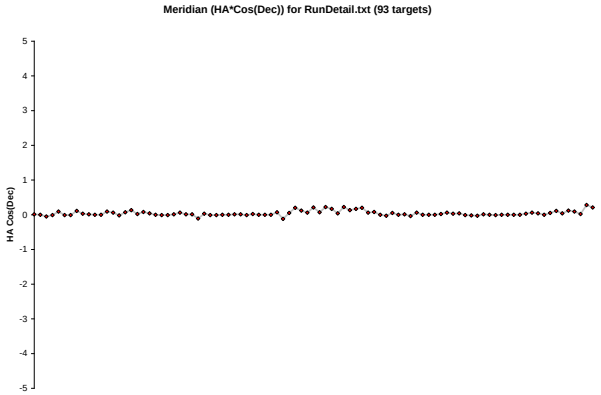

Figure 14. Meridian chart for 100% transit alt. (overload)

11.2.2 Lateness Only (over-loaded)

With only the Lateness term in the Efficiency function, the target sequence chart in Figure 15 shows a total preference for the earliest targets. In this case, more plans were run (106 versus 93), presumably because no early targets are lost, and the telescope stays to the west as much as possible.

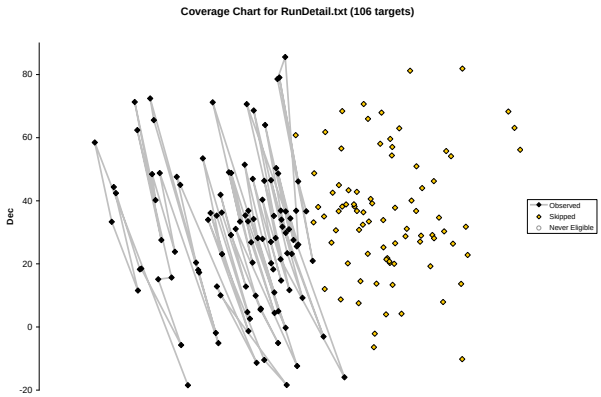

Figure 15. Target sequence for 100% lateness (overloaded)

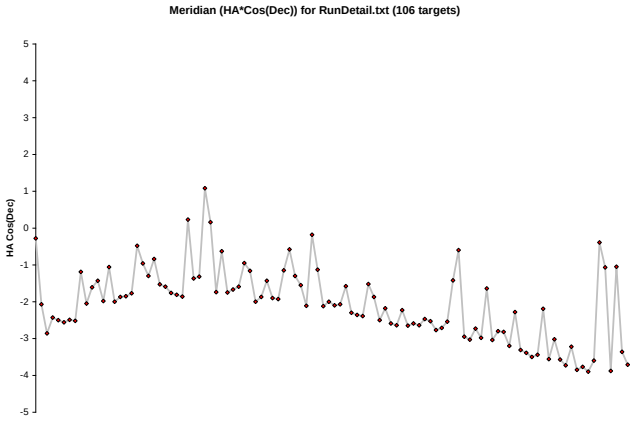

Figure 16. Meridian chart for 100% Lateness (overloaded)

11.3 Effect of Rising Plan Delay

Next, the effects of Rising Plan Delay were investigated at both moderate load (where its effects should be beneficial) and over-load (to see if it has detrimental effects). Priority and Slew Distance terms were disabled for clarity. The same sets of observing requests for moderate and overload cases were used.

11.3.1 Transit Alt. and Rising Plan Delay (moderate)

At a moderate load, the beneficial effects of Rising Plan Delay are clear when comparing Figure 18 with Figure 8 (same case but without Rising Plan Delay). All 69 possible plans were acquired. Most were acquired very close to the meridian, well above their constraints. Only early and late targets were (necessarily) acquired away from the meridian.

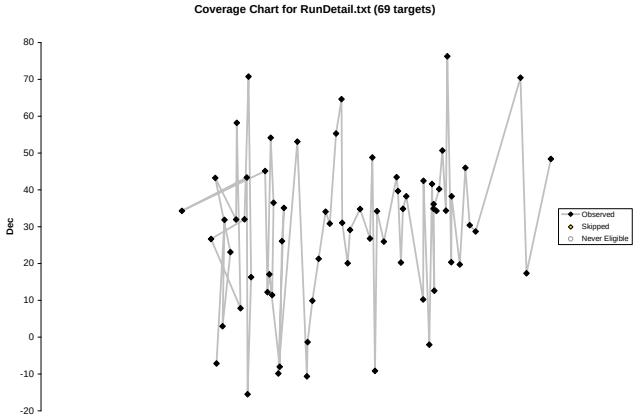

Figure 17. Target sequence for 100% Transit Altitude with Rising Plan Delay (moderate)

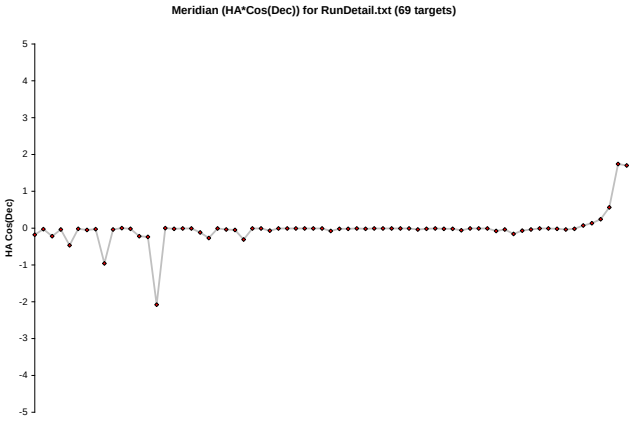

Figure 18. Meridian chart for 100% Transit Altitude with Rising Plan Delay (Moderate)

11.3.2 Lateness and Rising Plan Delay (moderate)

With Rising Plan Delay in effect, switching from 100% Transit Altitude to 100% Lateness caused very little change in the behavior of the dispatcher under moderate load. Again, all plans were completed. The only differences appeared at the beginning and end of the run, where the early targets were acquired earlier (and in a more favorable position) and some late targets were acquired in less favorable positions.

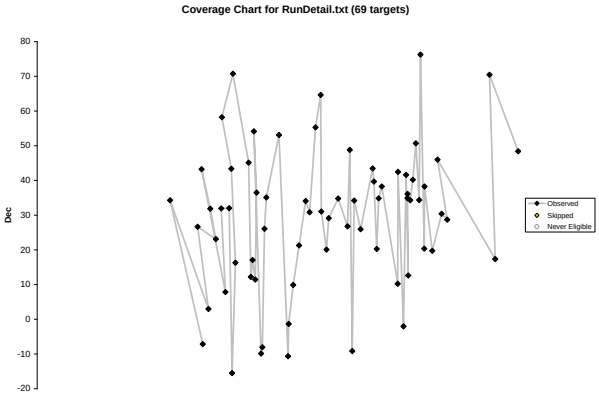

Figure 19. Target sequence for 100% Lateness with Rising Plan Delay (moderate)

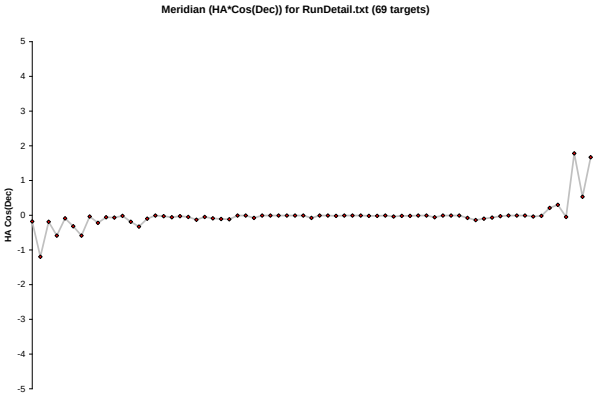

Figure 20. Meridian chart for 100% Lateness with Rising Plan Delay (moderate)

11.3.3 Transit Alt. and Rising Plan Delay (over-loaded)

Comparing Figure 22 with Figure 14 shows that the primary effect of Rising Plan Delay on an over-loaded schedule with Transit Altitude only is to allow more plans to be completed (109 versus 93), a benefit. Figure 22 shows a few deviations from the meridian through the run, but overall there seems to be no detrimental effect.

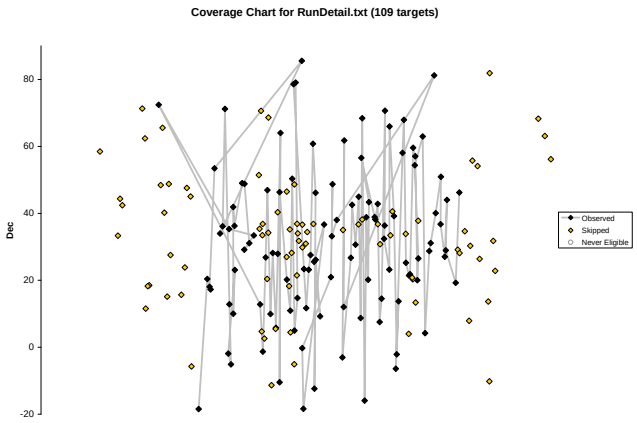

Figure 21. Target sequence for 100% Transit Altitude with Rising Plan Delay (over-loaded)

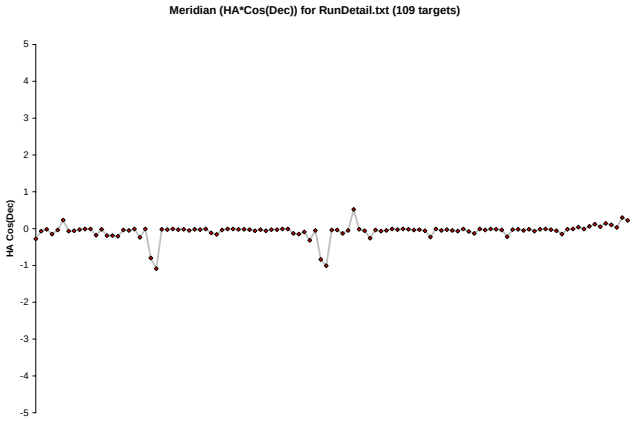

Figure 22. . Meridian chart for 100% Transit Altitude with Rising Plan Delay (over-loaded)

11.3.4 Lateness and Rising Plan Delay (over-loaded)

Comparing Figure 24 with Figure 16 shows that the effect of Rising Plan Delay on an over-loaded schedule with Lateness only is minimal. Again, more targets are acquired (though only a few, 108 versus 105), and there seem to be no detrimental effects. Figure 24 shows virtually the same target positions as Figure 16.

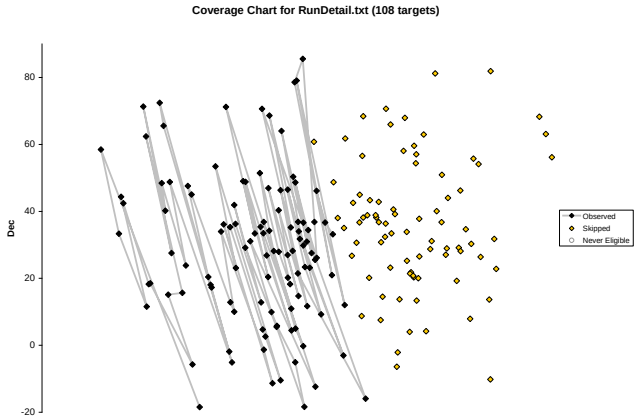

Figure 23. Target sequence for 100% Lateness with Rising Plan Delay (over-loaded)

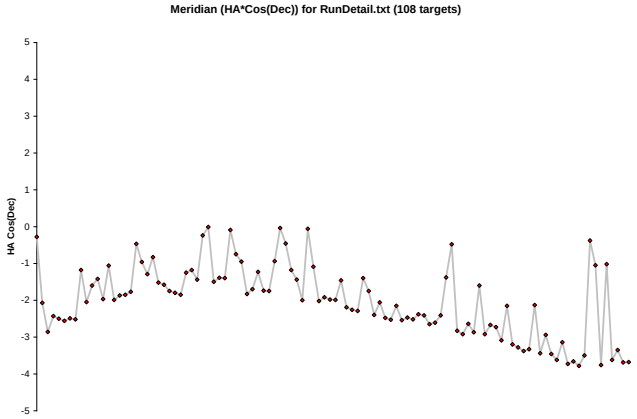

Figure 24. Meridian chart for 100% Lateness with Rising Plan Delay (over-loaded)

11.4 Combined Single and Quadruplets

The final set of simulations to be presented consists of combinations of plans containing random single images and plans each containing four linked observations (one 180-second image of each) of the same target (a simulated asteroid follow up). The linked observations were spaced 45 min apart with a 10 min. The dispatcher was configured to enable Rising Plan Delay and use only the Transit Altitude term in the Efficiency function. As before, other Efficiency terms such as priority were ignored for clarity. Again, the goal was to assure that Rising Plan Delay does not adversely impact the operation of the scheduler.

11.4.1 Combined Plans (moderately loaded)

The moderate load combined test consisted of 42 plans total, with 21 being random single images as previously described and 14 being simulated asteroid follow up plans with 4 linked observations as just described. Figure 26 shows that Rising Plan Delay is effective in preventing eastward drift of observing position as before. Of course there are deviations away from the meridian for the plans with linked observations. It is worth noting that, apart from early evening plans, the distribution of observing position about

the meridian is roughly symmetrical. This was not the case before the introduction of Rising Plan Delay.

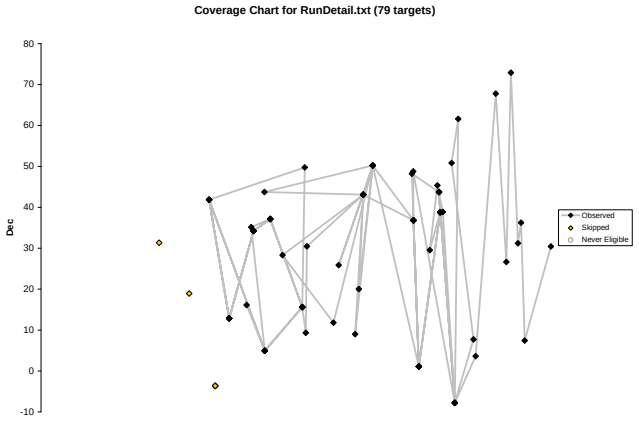

Figure 25. Target sequence for combined plans (moderate)

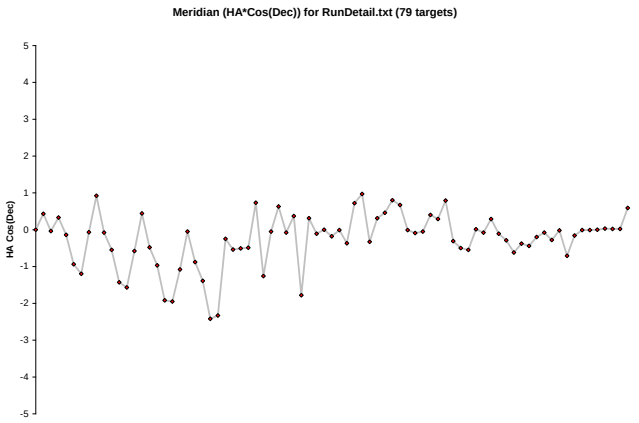

Figure 26. Meridian chart for combined plans (moderate)

12 Conclusions

The simulations results show that a dispatch scheduler is a practical (and in some ways, superior) alternative to queue-based optimizing schedulers. It has the following advantages:

1. Responds to changes in observing conditions and dispatches requests that *can* be done in the current (or worse) conditions while trying to do the ones that *must* be done in the current (or better) conditions.
2. May be interrupted by bad weather, and will resume by dispatching the best observations instead of merely delaying uncompleted ones (which will be west of their originally planned places).
3. Accepts new observing requests during the run and makes them immediately available.
4. Allows modification of unstarted requests at any time.
5. Will retry failed observations automatically.

It should be noted, however, that a dispatch scheduler is not applicable in all situations. At one end of the spectrum is a plan with many short exposures, such as used by a supernova search program. In this case, observing efficiency is paramount. The additional dispatch time between (very short) plans, and the randomness of the traversal sequence between targets, may significantly impact the total number of images that can be acquired in a run. For example, 3 additional seconds of dispatch time over 600 images reduces observing time by a half hour. At the other end of the spectrum is the astro-photography application in which a single target is imaged for long periods of time, up to the entire night. In this case, dynamic scheduling is probably unnecessary.

12.1 Rising Plan Delay

The behavior of the revised dispatcher with Rising Plan Delay conditions exceeded expectations. It seems clear from the evidence presented that Rising Plan Delay solves the "eastward drift" problem by improving the observing positions for light and moderate schedule booking levels, while having no adverse effects on scheduler behavior under over-booked conditions. More tests are needed, however, to validate "edge conditions" and to check for stability problems when recycling plans that have failed. For now, plans are not automatically recycled. Revision Status

The May 2006 paper is a general revision to the paper of the same title originally submitted for review and publication to the *Society for Astronomical Sciences* on March 30, 2004, and covers more detail as well as final design aspects at the time of commercial release (including the addition of the Rising Plan Delay feature).

The November 2014 revision (3) just corrected a few grammatical and typographic errors.

The January 2018 revision (4) includes information on the new Project Completion weighting factor for dispatch decisions. Also included is a description of the changes and additions to the test plan generator in support of simulating realistic workloads for astro-imaging and multi-night simulation.

The September 2018 revision (5) of this paper (September 2018) makes some small changes in the Rising Plan Delay algorithm, to prevent excessive imaging past (west of) the meridian.

13 Acknowledgements:

My heartfelt thanks go to Dr. Frederick Hessman of the University of Göttingen, Germany. Once he understood that I was looking at dispatch scheduling, he pointed me to the Steele and Carter paper [1]. In addition, he suggested the open-ended constraint design, including the idea of plug-in modules for constraints. This turned out to be an excellent suggestion.

In addition, John Farrell, formerly of Los Alamos National Laboratory and currently a director of the Las Cumbres Observatories, provided countless hours of testing simulations, and analysis. His analysis provided the impetus for the addition of the "lateness" term in the efficiency function, as well as providing insight for setting the standard

efficiency weights. More recently, he indicated the strong need for a change which resulted in the rising plan delay algorithm, as well as the "highest altitude" alternative to transit altitude in the efficiency function. My thanks go to him as well.

Finally, many of the changes and additions to the scheduler in its new version are due to the feedback of the scheduler users. In particular, the team of observers using the Sonoita Research Observatory (Walt Cooney, John Gross, Arne Henden, Dirk Terrell), as well as Steve Brady (a prolific CV observer), were the source of excellent feedback and suggestions. I truly appreciate their patience and enthusiasm.

References:

1. Steele I.A., Carter, D., 1997, *Control Software and Scheduling of the Liverpool Robotic Telescope* in *Telescope Control Systems II*, Proc. SPIE, **3112**, ed., H. Lewis, 222.
2. Puxley P., Boroson T., 1997, *Observing with a 21st Century Ground-Based Telescope* in *Optical Telescopes of Today and Tomorrow*, Proc. SPIE, **2871**, ed. Arne L. Ardeberg, 744

Contact Information

Robert B. Denny
DC-3 Dreams, SP
6665 E. Vanguard St.
Mesa, AZ 85215-7737 USA
+1 480 396 9700
rdenny@dc3.com