Gemini Automated Scheduler Software Design

By Sebastian Raaphorst, Bryan Miller, Arturo Nuñez, Sergio Troncoso, Kristin Chiboucas, Fredrik Rantakyro, German Gimeno, and Shane Walker

April, 2022

Table of Contents

1 Introduction

The automated Scheduler will be an essential element of the new Gemini Program Platform (GPP), which comprises upgrades and improvements to the Gemini Observatory Control System (OCS). The automated Scheduler is also one of the major components of the Gemini in the Era of Multi-Messenger Astronomy (GEMMA) program. GEMMA is funded by a special allocation from the US National Science Foundation (NSF).

A *Queue Plan* (henceforth referred to simply as a plan) is a collection of observations assigned to be done at specific times at designated sites. Traditionally, nightly queue plans per site are manually created by personnel at each site called *Queue Coordinators* (QC). Preparing these plans is a time-consuming process, as it must be done for different combinations of environmental conditions and other criteria. The goal of the automated Scheduler is to algorithmically generate plans in tandem for both Gemini sites. Automatically taking into account factors such as weather conditions and current telescope configurations (e.g. the instruments installed at each site, along with their components), the Scheduler will select viable observations for an observing period based on priority and create an optimized observing plan. This process can adapt as necessary to environmental changes, faults, or other factors, and makes it easier to manage observations for multiple sites in the same program.

This document describes the software design of a Scheduler service that meets the concepts and requirements described in the Scheduler Requirements document. The Scheduler will automatically schedule groups of observations within science programs in order to achieve the observatory's program completion goals.

In Section 2, we discuss the structure and concepts relevant to science programs at Gemini for the Scheduler. Section 3 describes the operational concepts behind the Scheduler, such as the data it will need to run and how it fits into Gemini's software architecture moving forward. We describe the architecture of the Scheduler in Section 4, including the different components that comprise the service, how they interact, how users interact with the Scheduler, and the algorithm that the Scheduler uses to produce nightly plans. Section 5 presents the various interfaces that the Scheduler uses to communicate with other Gemini services, and the interface that the Scheduler offers to allow access to the output it produces. Lastly, in Section 6, we discuss the technologies chosen that are used in the development of the Scheduler, and the testing and deployment strategies.

2 Science Program Concepts

We begin by detailing the concepts used in science programs that will be relevant to the automated Scheduler.

2.1 Program Organization

The components of a science program are described in the requirements document and the structure is outlined below. See Figure 1 for a graphical representation.

- Program level information (investigators, band, start and end dates, awarded time, active and completed flags, ToO status, etc.)
- Root Group: A top-level observation group (defined in Section 2.2) used in the organization of observations for the program. Observation groups form a hierarchy tree, and having a top-level group simplifies gathering all of the information about the program and its observations by traversing the tree.
    - Observation group
        - Subgroups or Observation (science or calibration)
            - Observation level information (constraints, targets, priority, etc.)
            - Observation Sequence (list of steps with instrument configurations, telescope offsets, etc.)
                - Atom (unsplittable group of steps)

The *observation* describes how to configure the telescopes and instruments to take data on a given location in the sky. Each observation is divided into individual *steps* that define the commands that will be carried out sequentially by the telescope and the instruments. This list of steps is called the *sequence*. Some steps of the sequence must be done together in order to produce a useful dataset. This minimum grouping of steps is known as an *atom,* or minimum schedulable unit. Atoms determine where sequences can be stopped or split when necessary.

2.2 Observation Groups

Observation Groups (or simply *groups*) are used to organize the observations in a program and capture any logical connections between them. There are two types of groups:

- AND groups: these are groups that usually require all subgroups (and thus observations in those subgroups) to be observed for the group to be considered complete. AND groups allow for additional structure to offer support for observations that must be done with timing or ordering requirements, and can be used to specify cadences.
- OR groups: these are groups that require `m` of `n` of their subgroups to be observed to be considered complete, where $1 \leq m < n$. For instance, OR groups are useful when an observation can be executed at either Gemini site, thus allowing for additional flexibility in the creation of plans.

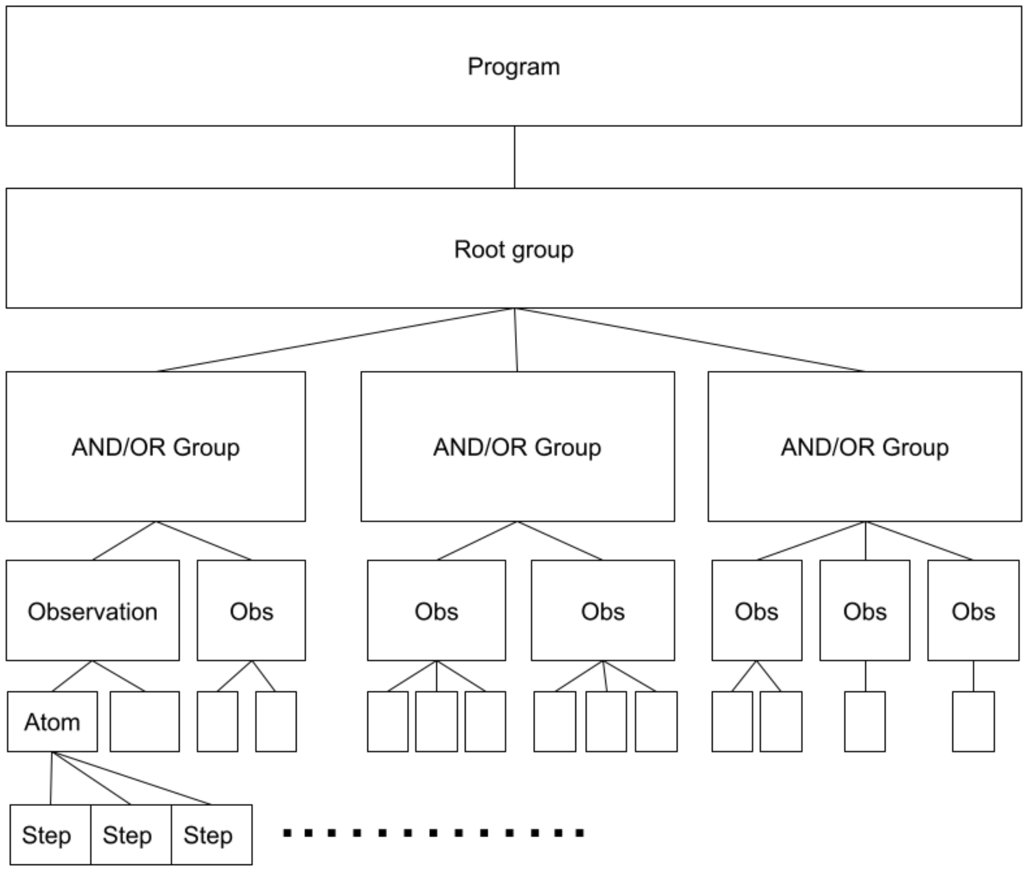


Figure 1: An example of the structure of a Gemini science program in GPP.

The current OCS offers limited support for AND groups and no support for OR groups. This is new functionality in GPP (see Section 3.7.1 of the GPP Inception Document) and the design presented here will be used to guide the design and implementation of groups in the other components of the GPP.

The parameters of the groups are specified via the GPP GUIs (e.g. Explore) or APIs, and they are used to organize the underlying observations for use by the Scheduler.

The Scheduler organizes the observations into *visits*, the largest block that the Scheduler has to fit into the plan. A *visit* consists of a period of time when executing a particular observation group: this may be composed of one or more slews, acquisitions, calibrations, and scientific data collections (typically observations). They are effectively internal AND groups but they can also be split when necessary.

Figure 2 shows the organization of different types of basic AND groups (consecutive, same night, and cadences, as described in Section 3.7.1 of the GPP Inception Design Document) and the resulting visits. Single observations not explicitly included in a group are given their own internal AND group and visit so that they can be treated equivalently to other groups.

Consecutive AND groups contain observations that must be done one after the other, either in the order given or in the most convenient or appropriate order for scheduling. The Scheduler needs to know the total length of the visit and how to order the observations.

Same night AND groups are for observations / visits that must be done on the same night. They are useful, for instance, for photometric standards that must be done on the same night as a science observation in order to calibrate it. These can be handled by setting relative timing constraints between visits (e.g. a time delay and window length after a given visit as described in Appendix A) and setting an internal timing window equal to the current night.

Cadence AND groups are for observations / visits that must be done in a defined consecutive order, usually over multiple nights. Relative timing constraints are used to specify the spacings and windows between the visits. More detail is given in Appendix A.

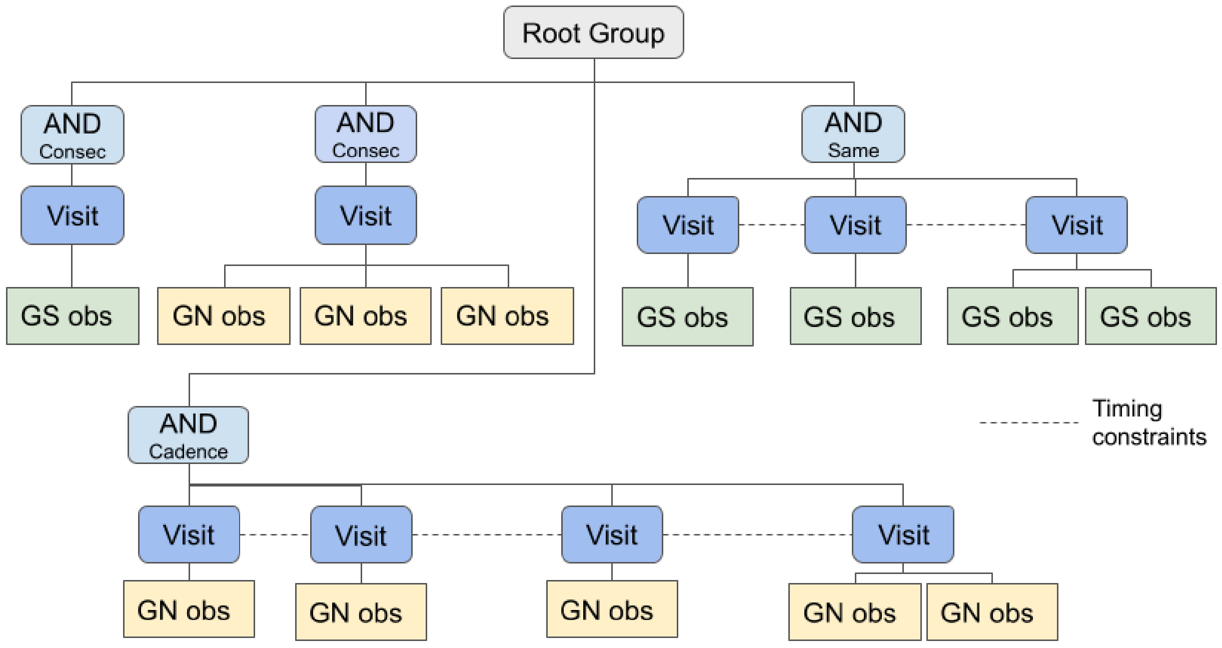


Figure 2: Examples of basic AND groups and visits within a program.

Examples of basic OR groups are given in Figure 3. These can be used to tell the Scheduler to do some subset of visits (`n` of `m`), which can provide more flexibility for scheduling when there are a lot of potential targets. OR groups also allow cross-site flexibility since one can define options to be executed at both Gemini North and Gemini South options if each site has equivalent capabilities. The OR group will keep track of what components are completed. Once the group requirements have been met, then the rest of the visits are effectively deactivated. During the scheduling process, the Scheduler tracks this provisionally.

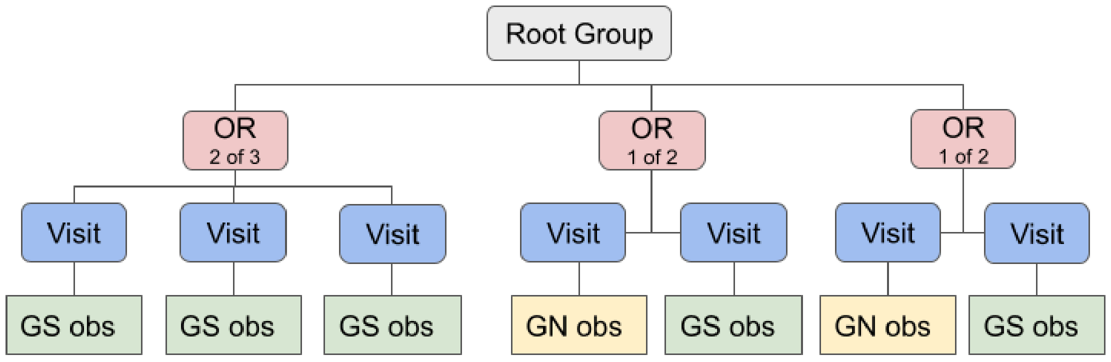


Figure 3: Examples of basic OR groups within a program.

2.3 Nightly Timeline

As stated in the science requirements, a *night* at a site is currently defined as the time between nautical twilights (the sun 12 degrees below the horizon), and science observations should only be scheduled during the official night. However, it is acceptable to do an acquisition before the evening nautical twilight in order to save the darker time for science exposures. Twilights are also used for standard stars (e.g. tellurics, photometric or spectrophotometric) and unguided calibrations such as flats. Occasionally, science observations must be done during brighter twilight periods because of the visibilities of the targets; therefore, the Scheduler creates plans between sunset and sunrise. A schematic timeline for a typical night is shown in Figure 4.

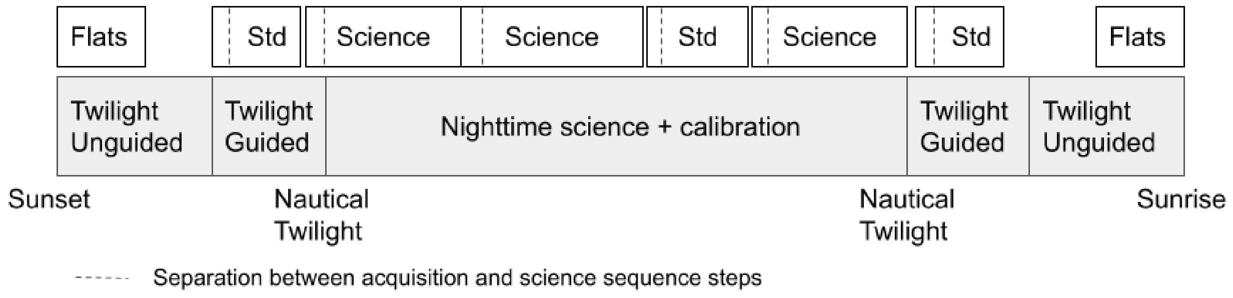


Figure 4: A schematic timeline for a typical night showing the major events that impact scheduling and the types of observations that can be scheduled at different times.

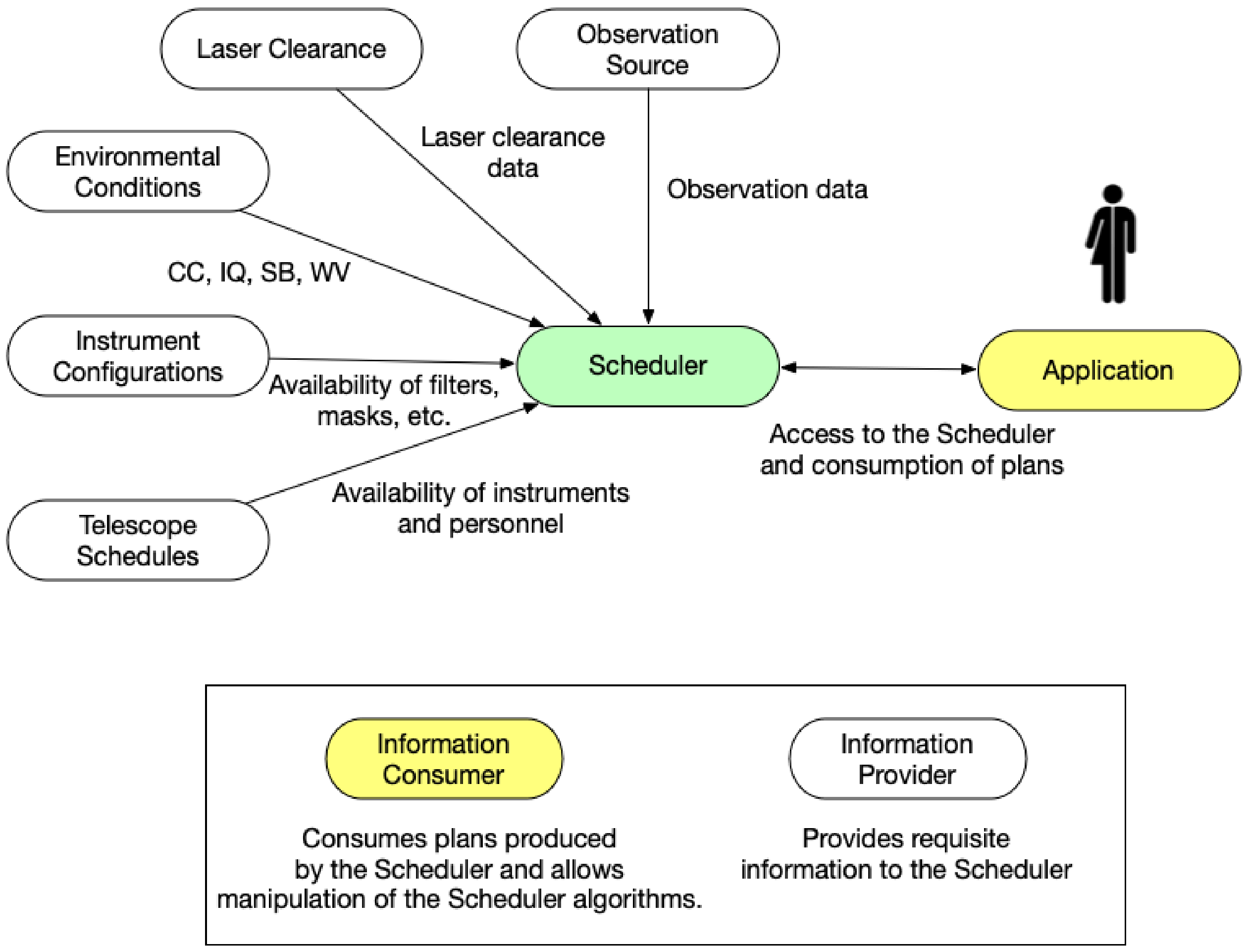
3 Scheduler Operational Concepts

The Scheduler can be executed in three modes:

1. **Validation mode**: A fixed input is provided to the Scheduler to obtain plans for past nights or full semesters and to compare both individual plans and the outcome of the Scheduler to what was actually achieved over a past night or semester. In addition, it also allows for testing changes to the service.

2. **Simulation mode**: This mode allows the Scheduler to facilitate planning, such as maintenance and component swaps. This mode also allows the Scheduler to create output in the form of long-term (e.g. up to a semester, which is a six month period) successive nightly plans for simulated weather conditions.
3. **Operations mode**: This is the mode where the Scheduler works in conjunction with other Gemini services to generate plans in real time during nighttime.

Figure 5 provides a context diagram of the scheduler.


Laser Clearance
Observation Source
Environmental Conditions
Laser clearance data
Observation data
CC, IQ, SB, WV
Instrument Configurations
Scheduler
Application
Availability of filters, masks, etc.
Access to the Scheduler and consumption of plans
Availability of instruments and personnel
Telescope Schedules
Information Consumer
Information Provider
Consumes plans produced by the Scheduler and allows manipulation of the Scheduler algorithms.
Provides requisite information to the Scheduler
Figure 5: Scheduler Context Diagram.

The Scheduler will employ the same APIs and protocols in use throughout other parts of the GPP (see Section 12 of the GPP Inception Document and Section 5 for more information). Each of these interactions is further detailed in the sections below.

3.1 Observation Source

The Scheduler requires a source of observations, which depends on what mode the Scheduler is running in: when in Validation or Simulation mode, observations are supplied by several possible providers - namely files or GPP services - containing simulation information, whereas when in Operations mode, observations are provided by the GPP Observing Database.

3.2 Environmental Conditions

In order to find which observations are eligible to be observed, weather forecasts and current conditions from a variety of sources are required. The main elements of this data include cloud cover (CC), image quality (IQ), sky background (SB), and water vapor (WV). Observations specify the worst set of conditions under which they can be executed. Unlike the current system, these conditions constraints will be specified as required at the position of the target rather than at zenith. The main difference will be in the handling of image quality. Models of how the constraints vary over the sky will be used to predict the values in other locations. An example is given in Section 4.2.2.

In Validation and Simulation mode, this data is supplied by lookup table files or GPP services such as a weather simulator, whereas in Operations mode, it will come from the GPP's Environmental service, Env.

3.3 Telescope Schedules and Instrument Configurations

This defines when various instruments and components will be connected to the telescope, as well as when different personnel will be available. Many instruments have configuration details such as filters, gratings, etc. that can be changed. As some observations require a specific instrument configuration or observing personnel, this information is used to ascertain if constraints regarding an observation are met to determine the set of observations that are viable to be scheduled during the specified time period.

In Validation or Simulation mode, the Scheduler can use data that is simulated or obtained from the GPP Resource API. Operations mode will rely on the GPP Resource API for this information.

3.4 Laser Clearance

Both Gemini telescopes are equipped with laser guide star (LGS) systems in order to perform adaptive optics. To use the telescope laser during an observation, Gemini requests approval from the U.S. Space Command's Laser Clearing House (LCH) to propagate the laser at a target on a given night. The laser clearance windows when the laser can be propagated for that target are returned by LCH and stored in the Laser Target Tracking System (LTTS) databases (one per site). This information is available to the Scheduler via the LTTS API as described in Section

5.5. Laser shuttering windows for a target are periods where the laser is not approved for use at the target's position.

The currently deployed version of the LTTS application is inside the Gemini firewall, so that the LTTS ensures that the laser is shuttered during any shuttering windows. As with other GPP services, LTTS will be moved to the cloud, and the current system will be modified to shutter the laser automatically if connection with LTTS is lost. When running in Validation or Simulation modes, we can use current or historical data from the GPP LTTS service or simulated data. In Operations mode, the GPP LTTS API is used.

3.5 Scheduler

This is the Scheduler service itself, which is described in more detail in Section 4.

3.6 Applications

This comprises the applications that observatory personnel will use to interact with and consume data provided by the Scheduler. There are two primary applications that interact with the Scheduler:

- Observe: this application is used during the night to view the queue plan from the Scheduler, to request alternate observations, to respond to suggestions from the Scheduler, and to execute the observations. Observe is outside the scope of this document.
- Schedule: this application is the QC interface to the Scheduler, and is used to plan engineering tasks, run simulations, and to modify generated plans as needed, or to trigger the Scheduler to immediately rerun and generate a new plan when necessary, e.g. when weather conditions change or a fault occurs. Additionally, it is possible to turn off the automated Scheduler to support the use of manual plans, e.g. in the case of a classical observer. See Section 4.3 for more detailed information about the Schedule application.

4 Scheduler Architecture

The Scheduler is a service that will interface with the other GPP services. Internally, the Scheduler consists of the pipeline of software components as shown in Figure 6.

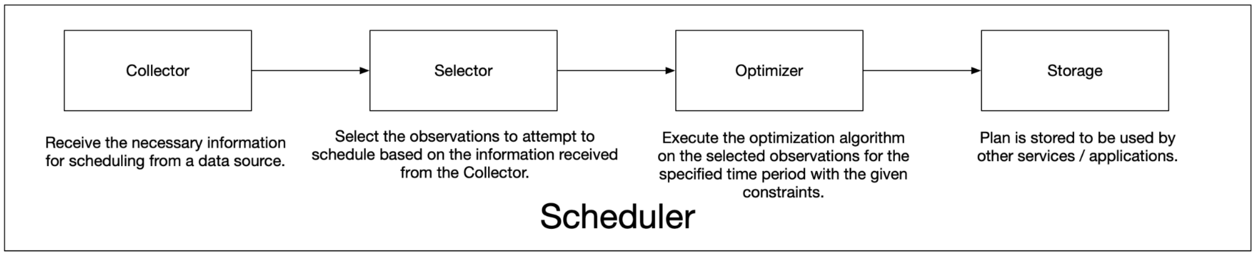

Collector
Selector
Optimizer
Storage
Receive the necessary information for scheduling from a data source.
Select the observations to attempt to schedule based on the information received from the Collector.
Execute the optimization algorithm on the selected observations for the specified time period with the given constraints.
Plan is stored to be used by other services / applications.
Scheduler

Figure 6: Scheduler Components.

Internally, the Scheduler contains a data model that comprises the necessary information from other GPP services in order to perform the required calculations that are used to produce a plan. This data model is detailed in Appendix B, and in Section 4.2, we describe in detail the Scheduler components and how the elements of the model are populated and used.

4.1 Scheduler Service

The Scheduler runs as a service and as per Section 3, supports three modes, i.e:

- An Operations mode that produces plans at night when requested, observations complete, science programs are modified or added, or environmental data or resource availability changes;
- A Simulation mode that runs on simulated data for planning or experimental purposes to see what effects resource or weather changes will have on the plan; and
- A Validation mode to run on historical data for testing performance, correctness, and plan quality.

In the case of Operations mode, exactly one instance of the Scheduler will run in the cloud alongside other GPP services. The Scheduler is notified of changes in data regarding science programs, environmental conditions, and available resources, and the Scheduler uses this information to determine if a request for a new plan should be triggered based on the information it receives. Plans are generated in real time and only the most recent plan is of interest. Because of this, only one task to produce a plan executes at any given time: if another task is running when a request for a new plan is triggered, the previous task is considered to be obsolete, is terminated, and the new task is instead executed. This is illustrated in Figure 7.

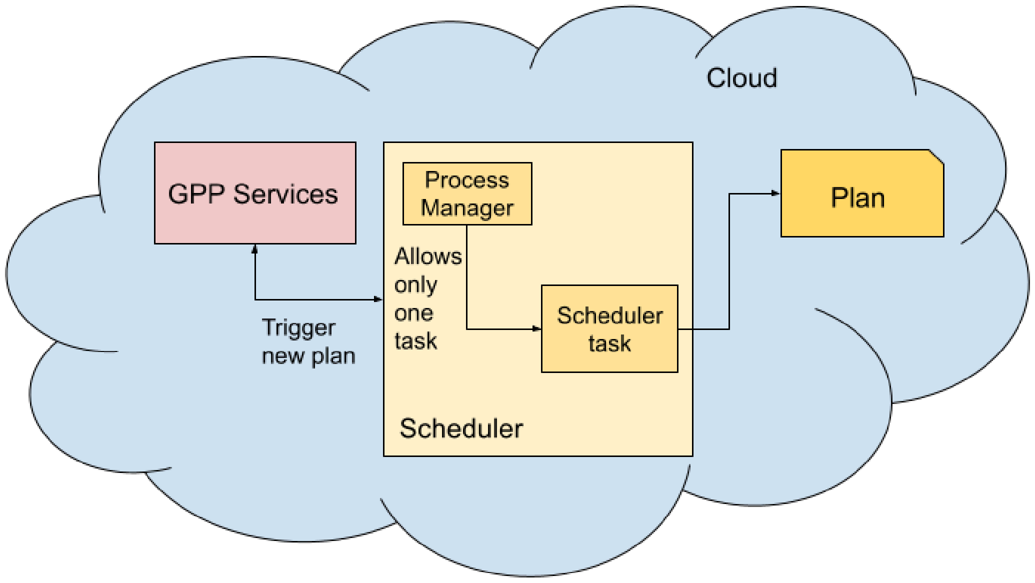


Figure 7: Deployment of the Scheduler in Operations mode. Note that a Scheduler task constitutes the execution of the Selector component through to the plan output.

In the case of Simulation or Validation modes, any number of instances of the Scheduler can be run either on the cloud or on a user machine, provided the machine has access to the necessary data for plan generation: this entails being able to access GPP services or having simulated data available. Since neither Simulation nor Validation results are required in real-time, execution requests are handled in parallel and run to completion. This is shown in Figure 8.

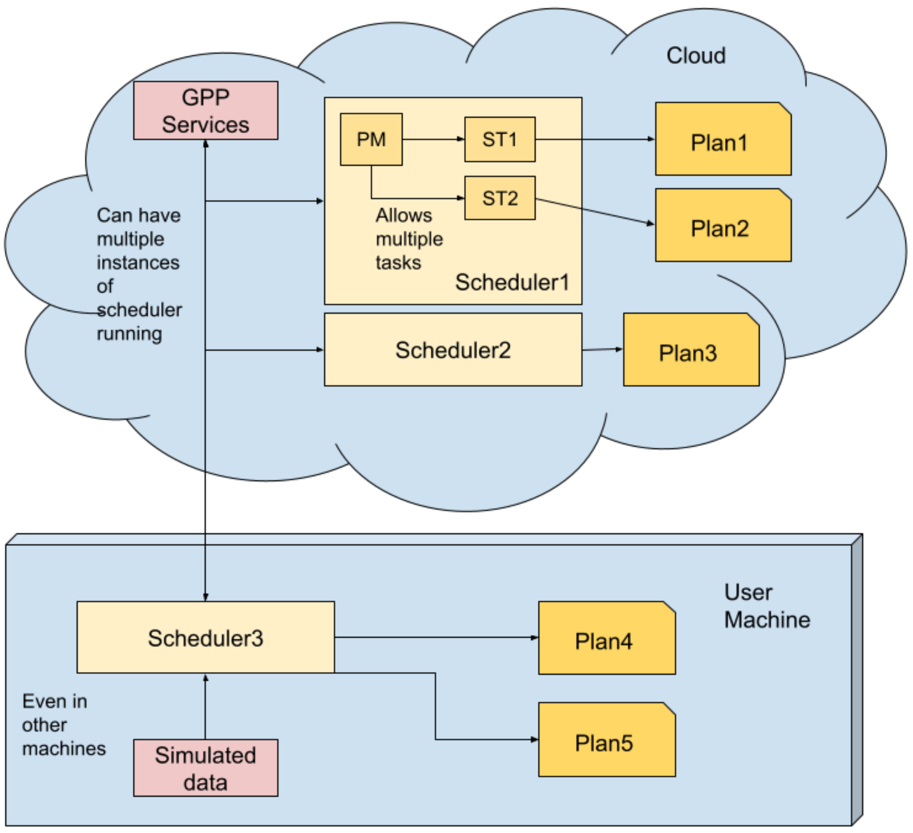


Figure 8: Deployment of the Scheduler in Simulation / Validation mode.

The mechanism that the Scheduler uses to handle these differing modes, to initiate new tasks, to perform any necessary administration on existing tasks, and to detect when a task to generate a new plan ends - whether it terminates, fails, or completes successfully - and to manage the associated output is performed by a *Process Manager*. Each running instance of the Scheduler has associated with it a Process Manager, which determines how that Scheduler instance responds to plan generation requests for the different Scheduler modes.

When the Scheduler is triggered to produce a new plan, the Process Manager has a handler called the *Runner* that is used to execute, manage, and resolve the outcome of that request. The Runner associates a *Scheduler Task* to a request and dispatches it: this Scheduler Task is an abstraction of a process that handles the plan generation request, such as start and end times for the plan and timeout values to terminate plan generation. The process handles the asynchronous flow that allows Scheduler Tasks to be independent from one another. Note that the Scheduler Task constitutes running the process from the Selector onwards to the output.

This process management structure offers the flexibility to:

- Run in real-time operations mode, limiting execution to only the most recently requested task at any given time, as illustrated in Figure 9; or
- Run in a multiprocess concurrent mode, where multiple tasks may be executed simultaneously as in purposes for Simulation or Validation mode, as shown in Figure 10.

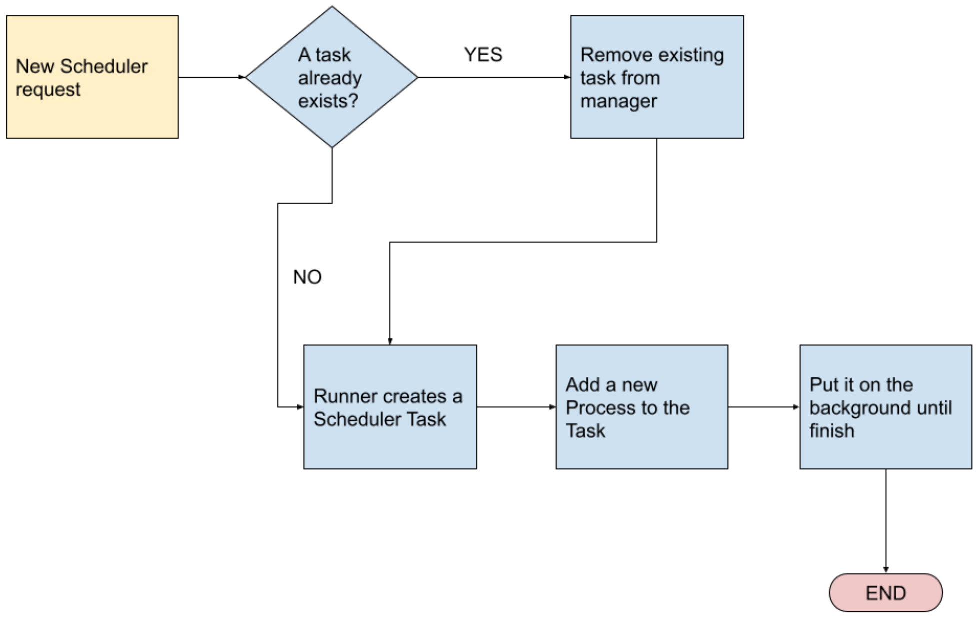

New Scheduler request
A task already exists?
YES
Remove existing task from manager
NO
Runner creates a Scheduler Task
Add a new Process to the Task
Put it on the background until finish
END

Figure 9: The real-time operations mode workflow of the Process Manager.

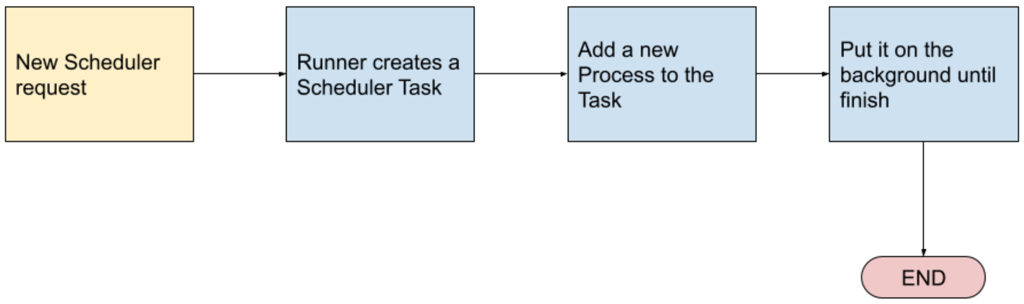

New Scheduler request
Runner creates a Scheduler Task
Add a new Process to the Task
Put it on the background until finish
END

Figure 10: The multiprocess simultaneous mode workflow of the Process Manager.

4.2 Scheduler Components

In this section, we describe the components that comprise the steps of an execution of a plan generation request to the Scheduler, as shown in Figure 6. These components are launched when the Scheduler receives a request, and are initialized with a time granularity for scheduling purposes, which determines the scale used in preparing plans: it represents the time length into which we want to break a scheduling period, e.g. if the granularity is one minute, we will break the scheduling period (e.g. the nightly timeline as described in Section 2.3 and shown in Figure 4) into consecutive blocks of one minute.

The Scheduler then gathers the necessary information to determine what observations are able to be scheduled for each time slot and determines a score representing the value of executing each observation at each step, which the Optimizer uses to prepare the plan.

4.2.1 Collector

Each running instance of the Scheduler has a single Collector, which is responsible for conglomerating a variety of information from other sources, predominantly:

- the current state of the executing observation;
- all observations that are ready for execution;
- environmental conditions;
- laser target clearance / shuttering windows;
- telescope instrument availability; and
- instrument configurations.

This is done using an API that populates the Scheduler model described in Appendix B.

Depending on the mode that Scheduler is running, the Collector allows for this information to be received from files, testing databases, or the operational GPP services. In the case of Operations mode, the Scheduler will subscribe to the necessary GPP services to receive notifications when changes occur (for instance, a new ToO), which the Scheduler will use to determine if it is necessary to trigger the calculation of a new plan. When files or testing services are being used, the Scheduler will automatically trigger the calculation of a new plan when simulation files or databases indicate that changes occur.

Additionally, the Collector requires pre-processed calculations that are necessary for the execution of the Scheduler but are independent of the content of the science programs themselves. These include, for the earliest program approved date to the last end date or for the period of a simulation, for each site of interest, and for a given time slot length, determining the following events for each night:

- Time of sunset and sunrise
- Time of evening and morning 12 degree (nautical) twilights
- Local midnight in UTC
- Solar position (both in ra / dec and alt / az) at each time slot
- Lunar position (both in ra / dec and alt / az) at each time slot
- Sun-moon angle at each time slot
- Illumination at each time slot
- Lunar distance at each time slot
- Lunar altitude at local midnight

As well, for each site and time slot, the pre-processor calculates UTC, local sidereal time, and local time.

This data is cached so that it is only calculated as needed. It can be populated by an external source, such as other GPP components, or it will be calculated by the Scheduler if it is not present when requested.

This information is stored in the **NightEvent** data object as detailed in Appendix B. The Collector stores this information so that it does not need to be recalculated for each site and night as it is costly to do so.

The observations that the Collector considers are only those that have a status of “Ongoing” or “Ready” (for a definition of these terms in Gemini’s context, see Section 3.3, Table 3.1 of the GPP inception design document). For each observation that is eligible, the Collector requires information about the target at the site of the observation for each night and time slot per night. This information can be provided directly to the Scheduler from other GPP components, or - if it is not present - the Collector is able to calculate and cache it:

- Coordinates of the target in ra / dec across the period of interest:
    - For non-sidereal targets, ephemerides data from HORIZONS queries are used and extrapolated across the time stots.
    - For sidereal targets, coordinates for a given epoch are used with proper motion data and the calculated time arrays.
- Altitude / azimuth
- Hour angle
- Airmass
- Parallactic angle
- Sky brightness assuming clear sky
- Visibility information, using elevation, timing window, sky brightness constraints, and guide star availability, which comprises:
    - A visibility array which indicates for what time slots the target is visible
    - The number of hours for which the target is visible during the night

This information is stored per observation and target in the **TargetInfo** data object as detailed in Appendix B.

The programs and observations populated into the Scheduler program data model, along with the calculated TargetInfo and NightInfo are passed to the Selector for further processing.

4.2.2 Selector

The Selector uses the data obtained and calculated by the Collector to determine which observations and observation groups are possible to perform for each time slot and assigns them a score representing their value to execute in that slot. This depends on the visibility of the observation target at a given site, how closely the required conditions for the observation match the environmental conditions for the time slot, and whether there are any restrictions on laser propagation for observations that require adaptive optics. This is done for each site.

The Selector deals with both visits and observations. It filters for visits whose observations meet the following criteria:

- The targets are visible (i.e. above the minimum elevation) during the night. The targets should be observable for enough time in the near term that there is a reasonable chance that the observation can be completed and the program can reach the completion goal.

- The resources necessary for an observation must be available.
- The required instruments and components are on the telescopes and available for use.
- The observations and programs have time remaining to be used.
- The actual conditions are at least as good or better than the observations' conditions constraints. If the conditions are better than necessary for an observation for a given time slot, the score for that observation is adjusted by a factor of the difference between the observation's condition constraints: this is done so that observations are preferably scheduled for time slots where the actual conditions are not far better than required.
- A guide star is available in the current or expected conditions. Also checked is when guide stars are available for nonsidereal or high proper motion targets and observations using parallactic angle mode.
- For observations using adaptive optics, there are laser clearance windows available.

The information and scores for observations are propagated upward through the group hierarchy to determine which groups can be performed, and to assign scores to them.

We provide an example below: the following table gives a set of observations available in the Collector with a minimum set of instrument configurations and conditions constraints. Note that FPU is a focal plane unit: either a slit or mask.

Inst	**Grating**	**FPU/Filter**	**Wavelength**	**IQ**	**CC**
GMOS-N	B600	IFU-2	0.5 um	1.2”@X=2.0(IQ70)	0.3 mag(CC70)
GMOS-N	R831	1.0arcsec	0.8 um	1.1”@X=1.8(IQ70)	0.1 mag(CC50)
GMOS-N	R400	Mask1	0.7 um	1.3”@X=2.0(IQ85)	0.3 mag(CC70)
GNIRS	321/mm	0.3arcsec	1.6 um	1.0”@X=1.5(IQ85)	1.0 mag(CC80)
GNIRS	1101/mm	0.45arcsec	1.6 um	0.9”@X=2.0(IQ70)	0.3 mag(CC70)
NIFS	K	IFU	2.2 um	0.8”@X=2.0(IQ70)	0.3 mag(CC70)
GMOS-S	B600	IFU-R	0.5 um	1.6”@X=2.0(IQ85)	0.3 mag(CC70)
GMOS-S	R150	Mask2	0.7 um	1.6”@X=2.0(IQ85)	1.0 mag(CC80)
GMOS-S	B1200	0.75arcsec	0.5 um	1.0”@X=2.0(IQ70)	0.1 mag(CC50)
F2	R3K	3pix	1.6 um	0.8”@X=1.6(IQ70)	0.3 mag(CC70)
F2	HK	Mask3	2.0 um	1.3”@X=2.0(IQ85)	1.0 mag(CC80)
F2	Img	K-short	2.1 um	0.5”@X=1.8(IQ20)	0.1 mag(CC50)

The Selector then uses the information received from the Collector regarding the available instrument configurations and the current conditions.

Available configurations:

Inst	Configuration
GMOS-N	B600 R400 R150 IFU-2 Mask1 1.0arcsec
GMOS-S	B600 R400 B1200 IFU-R 0.75arcsec
GNIRS	
NIRI	
F2	Mask3 3pix Img

Current conditions:

Site	IQ	CC	Wind
GN	0.9”@X=1.4(IQ70)/AZ50 in r	0.2 mag(CC70)	7 m/s@240 deg
GS	1.1”@X=1.2(IQ85)/AZ110 in g	0.05 mag(CC50)	4 m/s@330 deg

The following observations that match the available configurations and conditions are then selected. This subset is passed to the Optimizer for scheduling.

Inst	Grating	FPU/Filter	Conditions
GMOS-N	B600	IFU-2	IQ70 CC70
GNIRS	321/mm	0.3arcsec	IQ85 CC80
GNIRS	1101/mm	0.45arcsec	IQ70 CC70
GMOS-S	B600	IFU-R	IQ85 CC70
F2	HK	Mask3	IQ85 CC80

Measurements of conditions are used to predict the conditions at any location using sky models. For cloud cover, it is assumed that the mean extinction applies to the whole sky, unless there is enough data (temporally or spatially via all-sky cameras) to also calculate a variance. Image quality is modeled using either theoretical formulae, e.g. a power law, or empirical determinations of how IQ changes with elevation. The IQ at different wavelengths will be scaled using standard formulae or empirical measurements. See Figure 11 below.

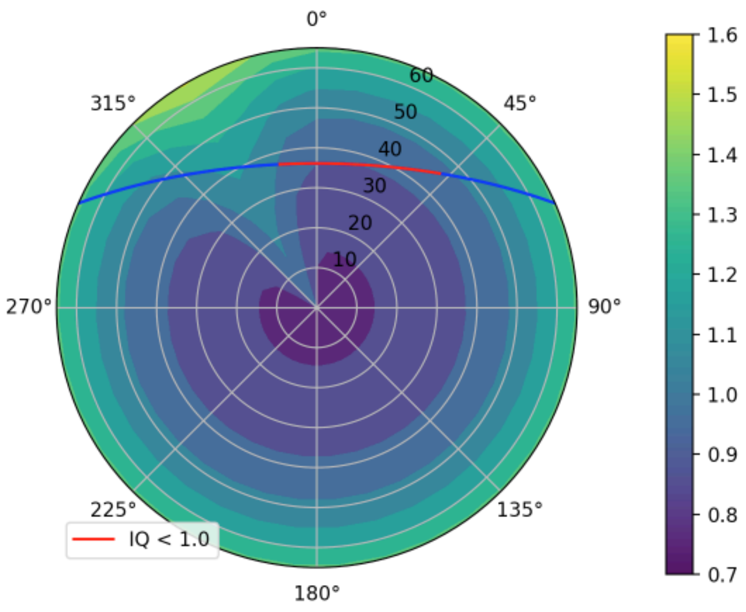


Figure 11: An example IQ model as a function of azimuth and zenith distance. The IQ changes with airmass according to the power law IQ ~ $X^{0.6}$ and a component for degradation due to wind has been added. A "measurement" of IQ=1.0 @ X=1.5 results in an IQ constraint of 1.0 arcsec being met for a target during the arc of its path across the sky (blue line) indicated in red.

This scoring algorithm and metric are explained in more detail in the Queue Scoring and Metric document. Using the information passed from the Collector, each observation is ranked with a score based on a number of factors such as:

- the band of a program (i.e. its priority as ranked by Gemini and its partners);
- a visibility fraction (i.e. the length of an observation or group divided by the total time it is available to be scheduled);
- visibility of the target for the remainder of the semester;
- whether or not the program containing the observation is for a thesis;
- the current completion percentage and how likely the program is to obtain completion;
- airmass;
- wind constraints;
- how well the observation matches the current environmental conditions;
- internal priority; and
- hour angle.

Note that this list is not exhaustive and further factors may be considered.

As described in Section 4.2.1, this process will be revisited multiple times. The Collector will trigger the Selector to repeat, and then re-filter and re-rank the observations when:

- environmental factors change or are forecast to change;
- observations are scheduled / completed;
- the planned time of an observation is inaccurate;

- faults are detected;
- additional programs / observations are added (generally in the form of ToOs);
- the user manually adjusts the score for one or more observations using the Schedule application or the API;
- the user specifically requests a new plan to be produced; or
- a certain amount of time has elapsed.

The Obsering Database service (see Section 5.1) notifies the Scheduler of Targets of Opportunity (ToO) as they are submitted. This triggers a recalculation of the plan. The Scheduler treats ToOs differently depending on their type:

- Interrupting: the Scheduler notifies the Observe service that a rapid ToO has come in, and advises that the current observation - if it is interruptible and the rapid ToO has sufficient allocated time to recompense the interrupted observation and be completed - should be interrupted and the rapid ToO observation should be done.
- Non-interrupting: no notification is given to the Observe service and the ToO is treated as a standard observation would be.

One of the goals of GPP is to treat ToOs as normal observations; their calculated score is typically increased by the ToOs having shorter timing windows, which in turn increases the likelihood of them being scheduled.

4.2.3 Optimizer

The Optimizer's task is to use the scores to place the observation visits into the Scheduler output, which consists of a series of nightly plans generated per site. The Scheduler output can range from the plan for a partial or single night to a series of plans for successive nights for up to six months. The Optimizer attempts to maximize the total score of the observations in each nightly plan, thereby assuring that the highest priority observations are completed, and fills each time period to the greatest extent possible, maximizing the telescope usage.

One of the primary goals of the GPP is to allow programs to be site-independent, so that observations can be run at either Gemini site (North or South) if their requirements are met. The Optimizer is designed to be capable of producing plans for any number of multiple sites, treating them as a network.

As mentioned in Section 2.3, the Optimizer will need to schedule the time between sunset and sunrise at each site. Constraints on guiding and sky background will result in calibrations being taken during twilights and guided science observations when the sky is darker (see Figure 4). Currently, time is only charged between the nautical twilights, when the sun is 12 degrees below the horizon, but the Optimizer will schedule science observations in twilights when needed.

To handle these cases, the Scheduler uses time arrays that stretch from sunset to sunrise for each night at each site to capture the variable lengths of nights across the year at different sites. Observations during twilights can then be handled using internal timing constraints, i.e. constraints added by the system rather than by the users. Unguided observations can be done at any time but twilight flats will be placed close to sunrise or sunset. Guided standards will be given more restricted internal timing windows such that they are done when guiding is possible or within some time (e.g. 20 minutes) before or after nautical twilight. The GPP Calibration service (see Section 5.4) is responsible for providing the Scheduler with the needed observations. Science observations will have a default internal timing window between nautical twilight. The Scheduler will have an internal rule that will allow the acquisition in twilight as long as guiding is possible. If a science observation must be done outside of nautical twilight, then either a timing window that includes the twilight can be set for the observation or instrument or wavelength-dependent rules can be defined that will set internal timing windows that include part of the twilights.

The Optimizer takes the scored observations and uses them to produce a plan for the night or other specified period of time. This process may be repeated many times, depending on changes to the pool of observations and their scores, or the amount of time taken to execute an observation, as detailed in Section 4.4.

4.2.4 Storage

The Storage function in the Scheduler is responsible for storing the output consisting of one or more successive nightly plans in a database to be consumed by other applications and services, like Observe, Schedule, or others.

The output will consist of entries of the form:

- Site
- Night
- Time slot
- Group to begin execution
- Atoms to be executed

An API for consumers to access the relevant information from Storage is offered via GraphQL.

In order to compare changes made to the Scheduler algorithms and the output produced to past output, data from the Collector is saved periodically or upon request internally to the Scheduler: the advantage to this is that the data is already in the form that the Optimizer requires.

4.3 Schedule Application

The Schedule application is the primary means with which QC interacts with the Scheduler. Early mockups of the Schedule application can be found in the figures in Appendix C.

Schedule allows QCs to select between plans generated from the automated Scheduler or manually generated plans, which are prepared using the current Queue Planning Tool (QPT). QCs can view the nightly plan for the selected site, including the scores for individual observations along with information about how those scores were derived. The total score for the automatically generated plan is shown, as are the scores for manual plans.

QCs are also able to tune the calculated scores of observations. These modified scores have an expiration date, but editing or deleting these is possible. These are displayed until the score changes are removed or expire; additionally, QC can adjust scores for all observations using the same systems / components (e.g. NIRI or R831, etc).

Schedule also offers a user interface for longer term planning and simulation testing: see Section 4 for details on running the automated Scheduler in Simulation mode. The user is required to specify the site of interest, conditions, rate of ToOs (and average ToO observation length, using mean and sigma), and the duration of the simulation. Default settings, conditions, and ToO rates are offered.

They must also choose between the actual semester telescope / instrument calendar or a simulated one in which they can make edits, which are able to be stored and selected for future use.

The simulator also makes use of a personnel calendar (using the GPP Resource service as discussed in Section 5.3) in order to know when to allow instrument component changes. The simulator is able to make decisions about when to implement component changes.

After running a simulation, the dates for that period are displayed along with the simulated conditions and number of ToOs for each night, the total plan score for the night, and the running completion rates for the different bands. Additionally, each active program in the database and the completion achieved by the end of the simulation is shown. The total metric score for the period, given this completion, is also displayed, along with other statistics - e.g. completion rates for each instrument.

It is possible to save each simulation and reload it to view it.

4.4 Validation, Simulation, and Operations Modes

As mentioned in Section 3, the Scheduler has three modes in which it can be run.

The Validation mode shown in Figure 12 is used for testing the Scheduler algorithm for correctness, performance, and to see the effects of different metrics / scoring functions. This will be done by running the Scheduler to generate output in the form of successive nightly plans for anywhere from a single night to across all nights in a semester. These will be used to verify the integrity of any modifications made to the scheduling algorithm to produce viable, sensible plans, and to compare plans produced using historical data or simulated data with those produced by either QCs or earlier versions of the Scheduler: plans over the course of a full past semester can be used to contrast the long-term performance of the Scheduler with what Gemini managed to accomplish during those semesters.

The Simulation mode shown in Figure 13 consists of running the Scheduler algorithm on current or simulated data to produce plans that can be used for scheduling purposes, looking ahead, or to simply generate plans for a night, over a period of 2 - 14 nights, or for a longer term. For example, a QC may be considering installing a different grating in GMOS. They can use the simulation mode to look at possible plans for different grating combinations given the range of likely conditions. They will choose the grating that gives the most options or the highest total score for the coming nights. This type of simulation can also be used to decide when to swap instruments throughout a semester.

In Operations mode - shown in Figure 14 - the Scheduler relies directly on GPP services to obtain the data needed to create plans:

- Current observation data will be collected from the Observing Database service;
- Current conditions will be collected from the Env service, which provides environmental monitors, forecasts, etc;
- Current instrument configurations and those planned for the future, as well as when instruments are available and specific personnel will be working, are provided by the Resource service.

As described in previous sections, the Scheduler service will subscribe to other GPP services to obtain this data, which is used to score observations, create output as a series of successive nightly plans, and then store it in a database for further use.

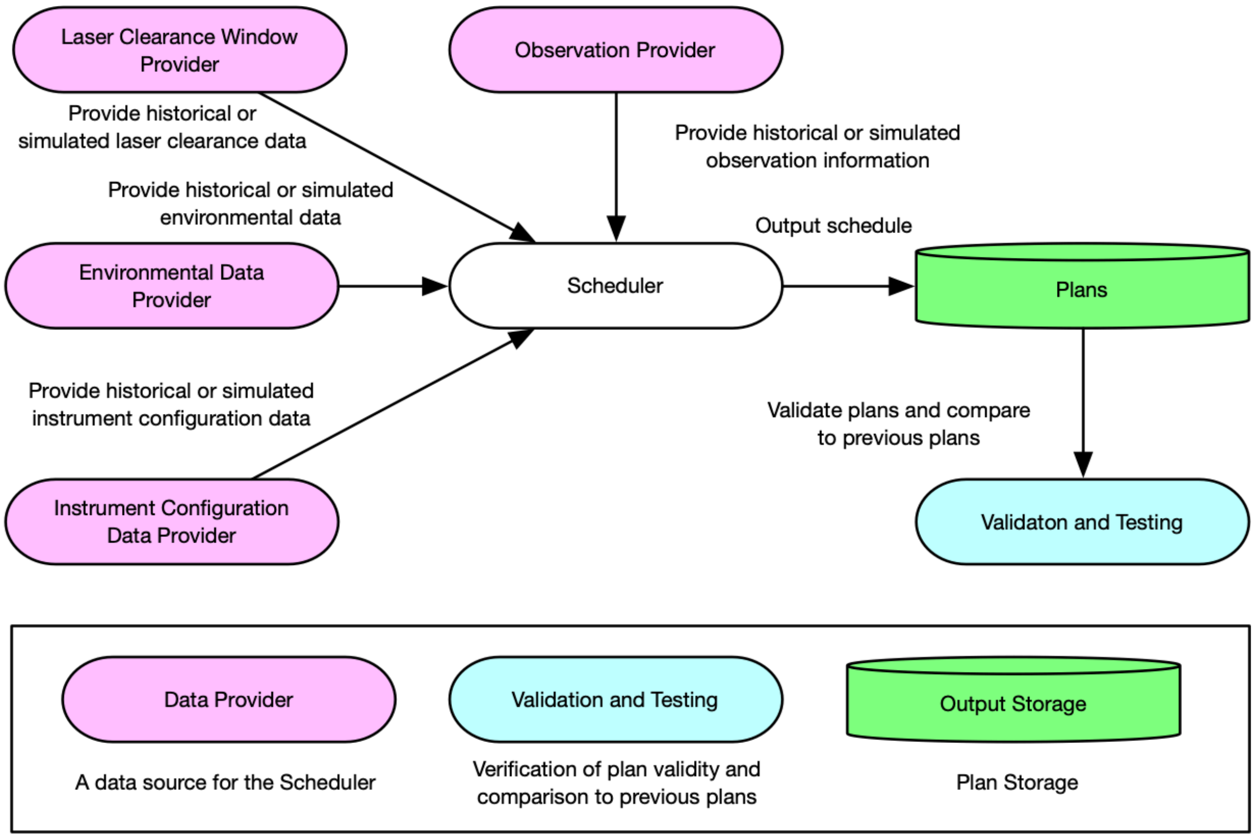

Laser Clearance Window Provider
Observation Provider
Provide historical or simulated laser clearance data
Provide historical or simulated observation information
Provide historical or simulated environmental data
Output schedule
Environmental Data Provider
Scheduler
Plans
Provide historical or simulated instrument configuration data
Validate plans and compare to previous plans
Instrument Configuration Data Provider
Validaton and Testing
Data Provider
Validation and Testing
Output Storage
A data source for the Scheduler
Verification of plan validity and comparison to previous plans
Plan Storage

Figure 12: Validation mode context diagram. The Scheduler will receive changes to input such as available resources and environmental conditions via the providers and cause the looping process as detailed in Section 4.2.2.

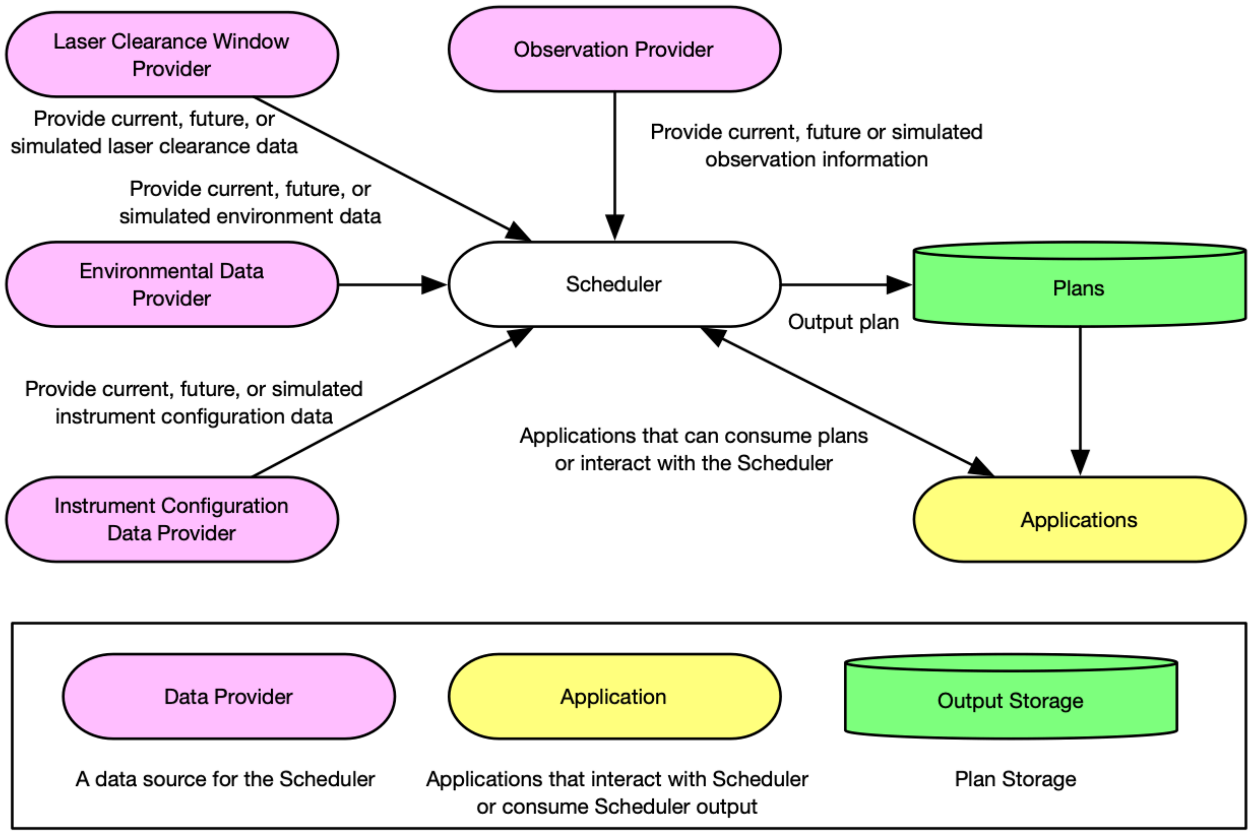

Laser Clearance Window Provider
Observation Provider
Provide current, future, or simulated laser clearance data
Provide current, future or simulated observation information
Provide current, future, or simulated environment data
Environmental Data Provider
Scheduler
Plans
Output plan
Provide current, future, or simulated instrument configuration data
Applications that can consume plans or interact with the Scheduler
Instrument Configuration Data Provider
Applications
Data Provider
Application
Output Storage
A data source for the Scheduler
Applications that interact with Scheduler or consume Scheduler output
Plan Storage

Figure 12: Simulation mode context diagram. Similar to Figure 11, the providers will inform the Scheduler as to changes in environmental conditions. One of the goals of Simulation mode will be to determine the best resource availability calendar for upcoming nights, as well as the priorities of remaining observations.

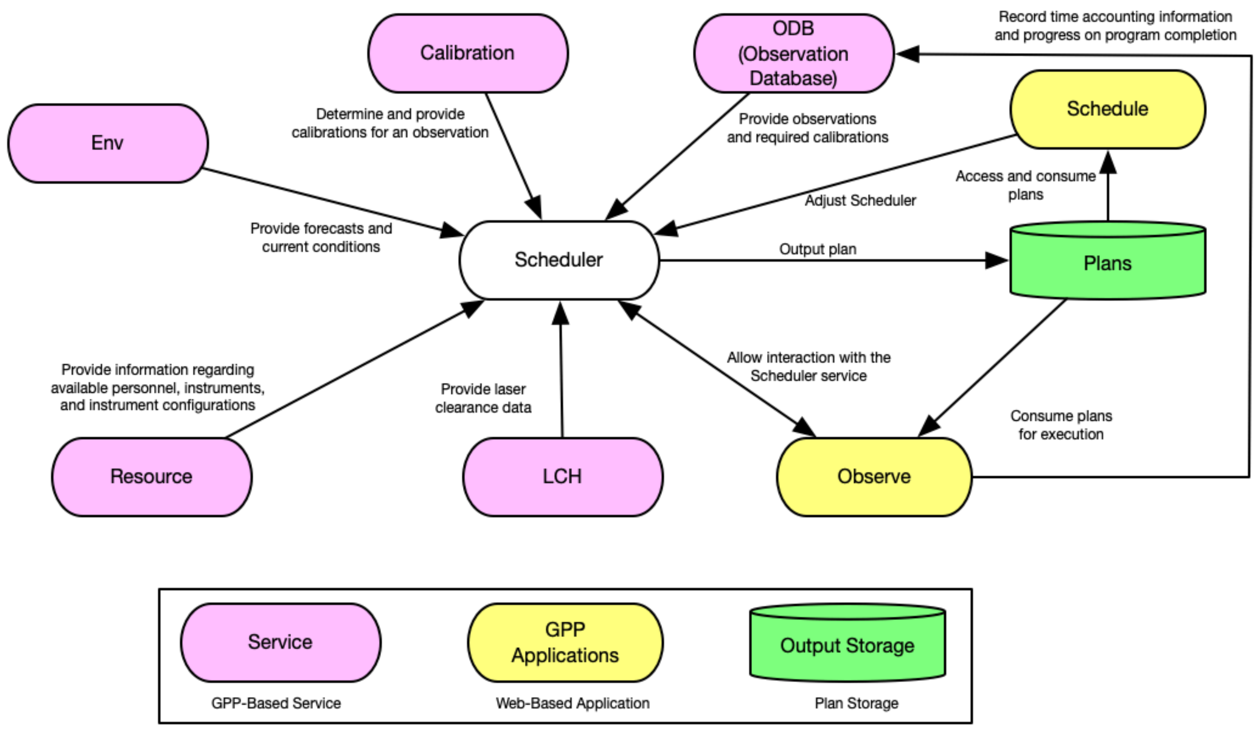

Calibration
ODB (Observation Database)
Record time accounting information and progress on program completion
Schedule
Env
Determine and provide calibrations for an observation
Provide observations and required calibrations
Access and consume plans
Adjust Scheduler
Provide forecasts and current conditions
Scheduler
Output plan
Plans
Allow interaction with the Scheduler service
Provide information regarding available personnel, instruments, and instrument configurations
Provide laser clearance data
Consume plans for execution
Resource
LCH
Observe
Service
GPP-Based Service
GPP Applications
Web-Based Application
Output Storage
Plan Storage

Figure 14: Operations mode context diagram.

Note that time accounting is an important consideration in the Scheduler. In general, the Scheduler assumes that everything that is added to the plan is observed; however, the amount of time that the Scheduler assumes an observation or some of its atoms will take to complete may deviate from the actual amount of time that it will take to achieve that completion. In all cases, the Scheduler is required to perform internal time accounting as it creates a plan in order to take factors such as remaining execution time and remaining awarded time into account. In the case of Operations mode, the official time accounting information will be done by the Observing Database (ODB) and stored, which will then be made available to the Scheduler when creating future plans. In the case of Simulation and Validation mode, since the plan is simply a simulation, it is necessary to rely on pseudo-time accounting, and assume that the estimated amount of time to perform an observation is the time it takes to perform an observation. A simulation manager or internal ODB will need to perform time accounting and feed the information back to the Collector. This results in a looping process as shown in Figure 15.

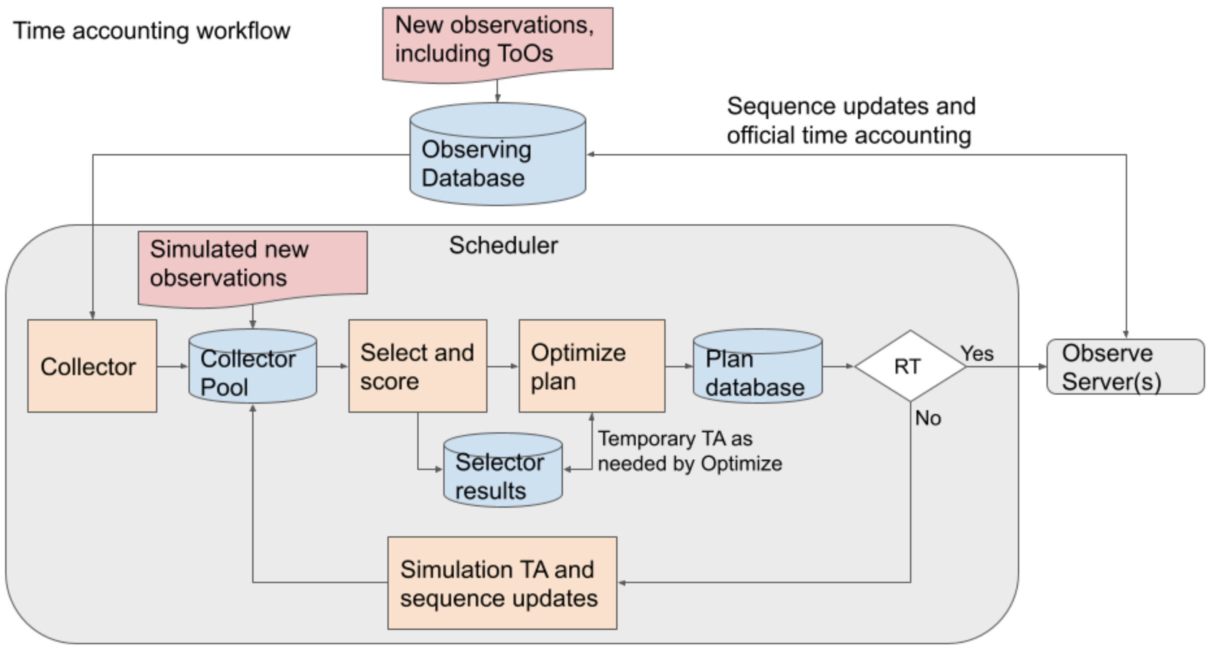

Time accounting workflow
New observations, including ToOs
Observing Database
Sequence updates and official time accounting
Scheduler
Simulated new observations
Collector
Collector Pool
Select and score
Optimize plan
Plan database
RT
Yes
No
Observe Server(s)
Selector results
Temporary TA as needed by Optimize
Simulation TA and sequence updates

Figure 15: Time accounting and looping within the Scheduler in the production of plans.

Each science program that is accepted is awarded time for execution on the telescope. As Gemini has multiple partners, the awarded time for a program may be from one participant, or spread across multiple participants (e.g. US, Korea, Argentina). The time is divided into different time accounting categories, with each category having access to different telescope resources.

The scheduling algorithm needs to keep track of the time used by observations and programs as they are added to the plan so that the Scheduler knows when to consider an observation, observation group, or program complete: this is in particular true when observations are split.

In general, the scheduling algorithm assumes that anything added to the plan will be "observed." These time accounting assumptions, however, cannot be directly applied to the observations in the GPP or the pool of observations selected as candidates for the plan since they have not yet been actually executed, and the actual time required may differ from the assumed time required. The information used for the scheduling algorithm, however, may be needed for subsequent plan generation.

For example, the Scheduler - running in Operations mode - might produce a plan at the start of the night that includes several observations from the same program. It is possible that the program has further observations, but they are unable to be added to the plan due to the program exhausting all of its allocated time from the observations that have already been scheduled. This process is highlighted in Section 4.2.3.

During the night, weather conditions may improve, thus triggering the generation of a new plan. In the new plan, the change to weather conditions may accommodate some of the aforementioned observations that had previously been omitted, which could lead to completion of the program.

Observation information stored in the GPP, which includes the steps performed and the time accounting, will be updated after atoms have been executed. In Operations mode, when this information is recorded in the GPP database, the Scheduler will be notified and the generation of a new plan with this data may be triggered if necessary.

In Simulation / Validation mode, however, since plans will not actually be executed and thus the information in the GPP will not be modified, to achieve a similar effect, a simulation manager will be needed to control this process instead. The Scheduler will require maintaining the starting set of candidate observations, which are then streamed through the Scheduler pipeline to produce a plan. The simulation manager will be required to keep track of simulation time, when new events occur, and what observations in the plans are considered executed. Instead of this information being updated in the GPP database, it is local to the Scheduler Simulation or Validation process, where the simulated time accounting occurs. As with Operations mode, when this information is updated, the process repeats.

We propose two options in order to implement the needed simulation manager:

1. The simulation manager contains a local copy of the observing database, which contains the rules for time accounting. This advantage to this approach is that it mirrors the structure of the GPP without affecting the data stored by the GPP. A risk to using this technique is that in Simulation or Validation mode, for each running task, a separate instance of the ODB must be used to segregate the data for each task.
2. The simulation manager feeds the time used in the plans directly to the Collector element (see Section 4.2.1), and that maintains the set of rules for time accounting. This is shown in the diagram in Figure 15. The advantage to this model is that it is simpler than maintaining an instance of the observing database inside the Scheduler, but the disadvantages are that it does not mirror the real-time Operations system as well, and that it also requires duplication time accounting rules between the observing database and the Scheduler, which requires careful management to avoid introducing inconsistencies into the GPP.

To accomplish the requirements detailed in this section, we selected a greedy algorithm we call GreedyMax, designed by Bryan Miller, to create plans that are at least as good as those that would be produced by a human QC (see Section 4.5 below). We have investigated other approaches, such as integer linear programming and genetic algorithms, with the results documented in the Gemini Automated Scheduler Trade Study document.

4.5 GreedyMax Algorithmic Details

The GreedyMax algorithm is a specific instantiation of the Optimizer in Figure 6. In this section, we provide a high-level description of how GreedyMax functions.

A *visit* consists of a period of time when executing a particular observation group: this may be composed of one or more slews, acquisitions, calibrations, and scientific data collections (typically observations). In general, a goal of the Optimizer is, given the observations selected by the Selector, to maximize the amount of time taking science data and to minimize the amount of overhead, including acquisitions. GreedyMax is not mathematically maximizing any function, so it cannot naturally maximize the science time; instead, it is a heuristic algorithm and must be given rules for how to split observations. Therefore, we define the concept of a *minimum visit length* (MVL) to avoid a plan that switches between many short visits that would be inefficient and have high overheads. We currently use a value of 30 minutes. For observations longer than the MVL the algorithm ensures that it will not produce pieces of observations that are less than this length. The MVL will be adjusted based on experiments with the Simulation and Validation modes and it can be dependent on the instrument, mode, and other considerations.

The pseudocode for the GreedyMax algorithm is described below, and illustrated via a flowchart in Figure 15.

- For each site in the set of sites (in Gemini's case, North and South):
    - Consider the next (chronologically) open or unscheduled time interval.
        - Loop over the remaining visits from the Selector and select the one with the maximum score in the interval.
            - Schedule as much of each visit as possible, splitting long visits and observations when necessary and trying to avoid a lot of short remnants. The piece of a visit that the algorithm is trying to schedule corresponds to the notion of a visit as defined above. Visits can be split at the boundaries of the observation sequence atoms. A new acquisition overhead is added to the remaining part of a split observation.
            - The visit is scheduled based on the integral of the weighting function over the minimum of the total time needed to complete the visit and the current time interval.
            - If the visit is shorter than the MVL, then schedule the entire visit.
            - If a visit is within the MVL of the edge of the available interval (including twilights), then attempt to push the visit to the boundary to avoid small gaps.
            - If there are calibrations in the visit, the number of standard stars (i.e. baseline calibrations) is determined by the scheduled length of the science observations. If only one is needed, then the standard is placed either before or after the science observations such that the difference in mean airmasses between the standard and science is minimized.
            - If the visit is part of a "same night" AND group, then the schedulability of the rest of the group will be evaluated. If feasible, then the remaining observations are placed or the scores are set

very high so that they can be placed in subsequent iterations. If the other visits cannot be placed, then the visit is ignored.

- If the visit is part of a cadence, then the timing window for the next visit in the cadence is set (see an example in Appendix D).
- If the visit is part of an OR group whose success criteria are met, then the scores for the unselected branch(es) of the group are deactivated (e.g. scores set to 0).
- The scheduled time for each observation and visit is recorded internally (without updating the Collector or ODB): thus, the Scheduler will know when to stop considering them. While the actual time accounting is handled by other GPP services, the Scheduler must perform some pseudo-time accounting to determine how much time is used by an observation or program so that it knows whether to stop scheduling those.

GreedyMax will first schedule the official nighttime between nautical twilights and then fill in the twilights based on the science observations scheduled and additional calibrations suggested by the calibration service (see Figure 4). Appendix D contains an example of how GreedyMax builds up a plan for a night.

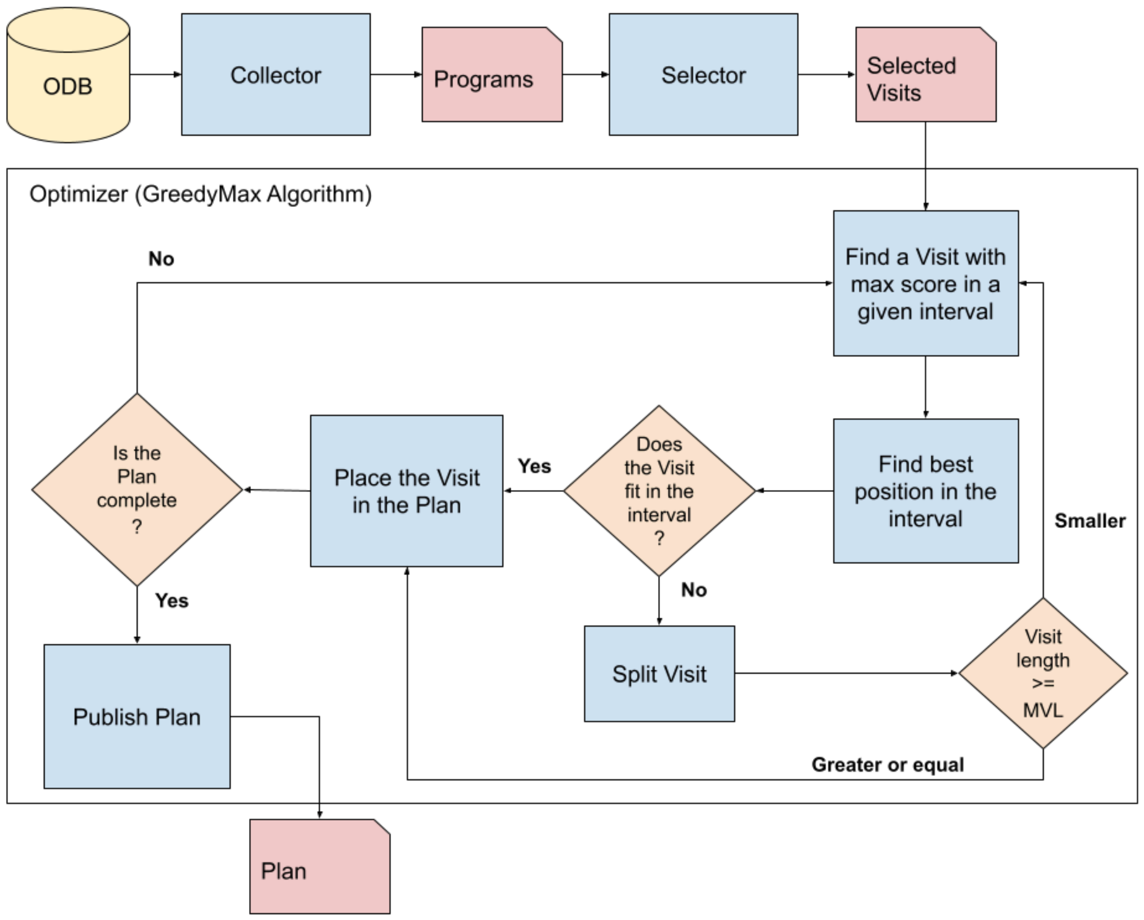

ODB
Collector
Programs
Selector
Selected Visits
Optimizer (GreedyMax Algorithm)
Find a Visit with max score in a given interval
No
Find best position in the interval
Does the Visit fit in the interval ?
Yes
Place the Visit in the Plan
Is the Plan complete ?
Yes
No
Smaller
Split Visit
Visit length >= MVL
Greater or equal
Publish Plan
Plan

Figure 15: A flowchart of the GreedyMax algorithm.

5 Software Interfaces

Here we list the software interfaces between the Scheduler and the main software components it interacts with.

5.1 Scheduler to Observing Database (ODB) Interface

The Observing Database service contains the science programs and their observations down to the level of detailed step configurations. The Scheduler needs most of this information including:

- Program attributes that contribute to scoring such as whether the research is part of a thesis investigation, the assigned science band, etc.
- Planned time estimates for observations and their individual steps
- Time accounting and awarded time
- Observation status, targets, conditions constraints, instrument resources required, GCAL (calibration unit with arc / flat lamps) configurations, timing windows, guide star availability, overheads, etc.

- Notifications when new observations are added (e.g. for Target of Opportunity observations)

5.2 Scheduler to Env Interface

The Env service will use conditions and forecasts from a variety of sources at a specific site to provide to the Scheduler. This information will be used to ensure that an observation's weather requirements are met, and in creating a score for an observation. Env will be provided with the following information:

- Site(s) of interest
- Datetime or range of datetimes
- Duration of interest
- Azimuth/Elevation of interest (optional)
- Constraint of interest (optional, return the full set of constraints if not given)

The Env service returns a list of the desired weather conditions for the site during the times requested if possible: a date too far in the future that cannot be statistically or accurately predicted returns a value of none.

If a sky location is not specified, then the conditions are returned for zenith, or preferably, as an all-sky model or map as shown in Figure 11. The details of the Env service are beyond the scope of this document but it uses all available information - including historical records, current sensor readings and pipeline results, observer input, and weather forecasts - to calculate the most likely conditions values and uncertainties.

5.3 Scheduler to Resource Interface

The Resource service provides the Scheduler with the information about the availability of the telescopes, their instruments, instrument components and modes, and personnel planned for a given site. It also can be used to manage changeable components such as slits, masks, gratings, and filters. The Resource service will be updated by nighttime staff if any significant faults occur that cause systems to become unavailable. This information will be propagated to the Scheduler via the Collector, and, if pertinent, trigger rerunning the scheduling algorithms.

The Scheduler will query Resource providing:

- Site(s) of interest
- Datetime or range of datetimes
- Duration of interest
- Instrument desired (as necessary)
- Required components and modes (as necessary)
- Required personnel (as necessary)

The Resource service should reply with:

- Available site(s)

- Datetimes
- Durations
- Available instruments
- Available components and modes
- Observing mode (queue, classical, priority visitor, engineering, etc) and programs that should receive priority
- Available personnel

5.4 Scheduler to Calibration Interface

The Calibration service will provide information about necessary calibrations and configurations for an observation, which are used by the Scheduler to determine what to include in observation groups. Some of this information may not be able to be predicted by the ODB service due to observation splitting and dependencies on when an observation can be scheduled.

The user can specify calibration choices if they so choose, in which case the Scheduler will use them, but by default, the Scheduler will pick the best star(s) based on when the observation is scheduled and the length of the visit. Depending on the instrument and mode, these include standard stars for telluric, radial velocity, PSF, photometric, and spectrophotometric calibrations.

The Scheduler will provide the Calibration service with the following information:

- Instrument
- Instrument mode or configuration
- RA (optional)
- Dec (optional)
- Duration (optional)
- Spectral types (optional)
- Calibration type (optional)

The Calibration service will reply with:

- A list of calibration candidates or full observations

5.5 Scheduler to LTTS Interface

This is motivated by the requirements for Gemini's laser system as described in Section 3.4. The Scheduler must not only obtain the laser shuttering windows, but also calculate the effective time over the observation for which data can be taken. This means that the actual shuttering windows need to be corrected with the overheads of opening the laser and adaptive optics (AO) loops, reading out the detector, closing the laser and AO loops, and then recording data.

The existing Laser Target Tracking System (LTTS) application will be adapted to provide an API layer to be used by other services. The Scheduler will send:

- The target of interest and its coordinates

- Starting datetime
- Duration

The LTTS service will reply with:

- Laser shuttering windows for the target from the starting datetime through to the duration

5.6 Scheduler to Applications - Observe Interface

Plans produced by the Scheduler are made available via query or subscription. The primary consumer of plans is the Observe application, which maintains a subscription to the Scheduler service to be informed as to the Scheduler's recommendation for the night's observing plan. The Scheduler will send a plan in the format described in Section 4.2.4, consisting of:

- Site
- Night
- Time slot
- Group to begin execution
- Atoms to be executed

This information is sufficient for Observe to query other GPP services such as the ODB to determine the necessary details to execute the atom steps.

5.7 Scheduler Interface

The Scheduler API will permit such operations as:

- Allowing the QC to tweak observation scores
- Fixing an observation for scheduling
- Defining time periods that the scheduler should not fill
- Recalculating a new plan
- Changing an observation's score
- Retrieving a plan
- Turning on or off the automated Scheduler

The Schedule application - as described in Section 4.3 - will serve as the primary means of manipulating the Scheduler service, although the API will be independently accessible.

The Scheduler can be turned off, and manual plans as prepared by a QC using the current QPT may be used instead.

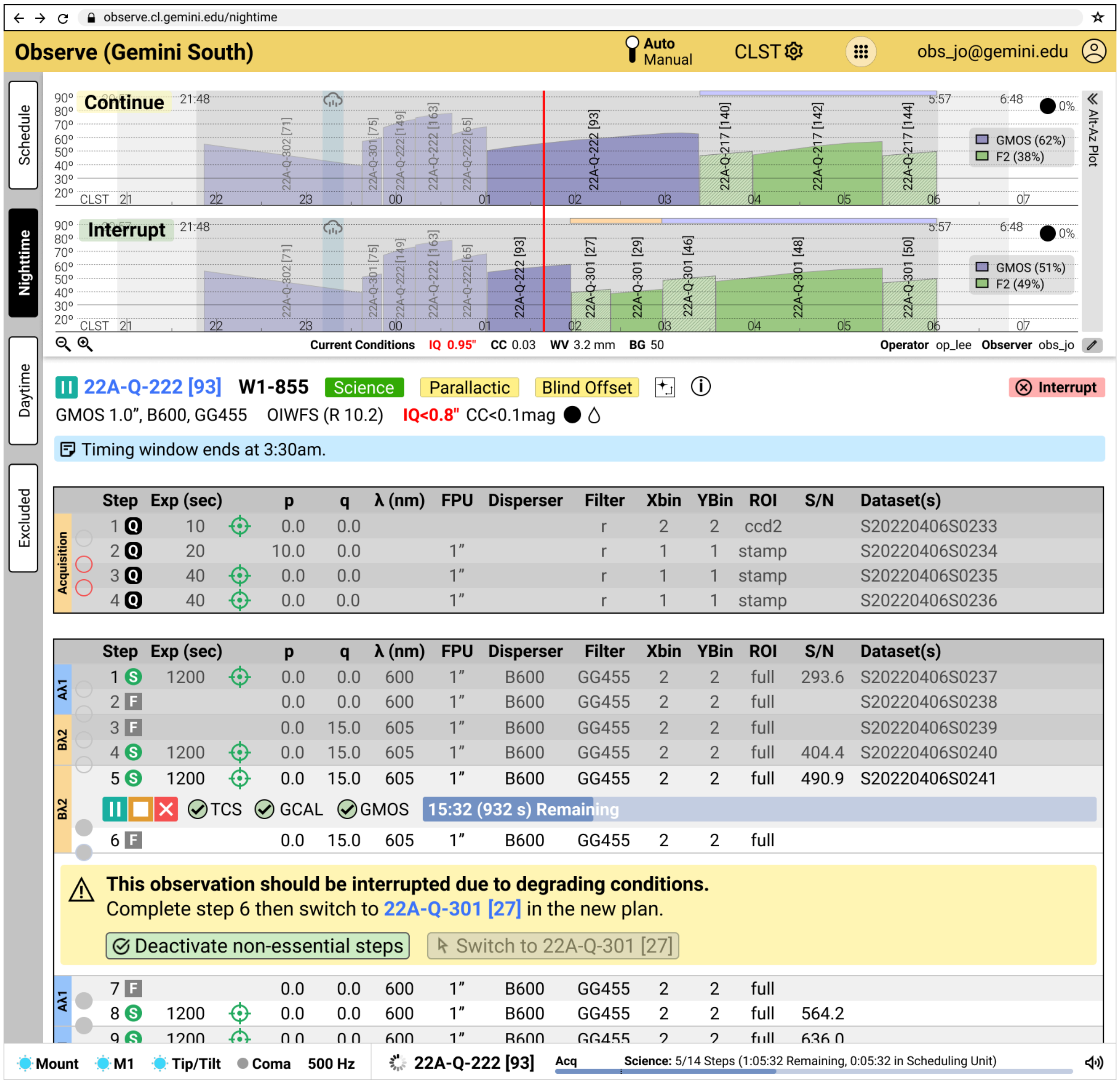
The Scheduler's output, i.e. a plan or series of successive nightly plans for sites of interest, will primarily be consumed by the GPP Observe application, which will subscribe to the Scheduler service to be notified of any changes to plans or creation of new plans. A plan is represented in the API as a list of allocations, where each allocation references an observation (or part of an observation) along with expected start and end times. Observe requires a subscription interface to obtain:

- The current status, i.e. whether it is calculating a plan update or idle

- The current plan as it changes over time
- Plan change guidance

The last point refers to observing assistance provided by the Scheduler. For example, when weather conditions change, the observer always has the option of switching to a new plan or continuing what they are currently doing. To help inform and guide the observer to make the best decision, the Scheduler should send a notification with the updated plan, a recommendation for wrapping up the current visit, and a new potential plan.

Figure 16 below presents an Observe mockup. The scheduler has just suggested a plan change due to degrading conditions. It shows the current plan (marked “Continue”), the plan that would be followed should the observer switch (marked “Interrupt”), and a recommendation for how to proceed. In particular the guidance is to “Complete step 6 then switch to 22A-Q-301 [27].”



22A-Q-222 [93] W1-855 Science Parallactic Blind Offset

GMOS 1.0", B600, GG455 OIWFS (R 10.2) IQ<0.8" CC<0.1mag

Timing window ends at 3:30am.

	Step	Exp (sec)		p	q	λ (nm)	FPU	Disperser	Filter	Xbin	YBin	ROI	S/N	Dataset(s)
Acquisition	1 Q	10	⌖	0.0	0.0				r	2	2	ccd2		S20220406S0233
	2 Q	20		10.0	0.0		1”		r	1	1	stamp		S20220406S0234
	3 Q	40	⌖	0.0	0.0		1”		r	1	1	stamp		S20220406S0235
	4 Q	40	⌖	0.0	0.0		1”		r	1	1	stamp		S20220406S0236

	Step	Exp (sec)		p	q	λ (nm)	FPU	Disperser	Filter	Xbin	YBin	ROI	S/N	Dataset(s)
AA1	1 S	1200	⌖	0.0	0.0	600	1”	B600	GG455	2	2	full	293.6	S20220406S0237
	2 F			0.0	0.0	600	1”	B600	GG455	2	2	full		S20220406S0238
BA2	3 F			0.0	15.0	605	1”	B600	GG455	2	2	full		S20220406S0239
	4 S	1200	⌖	0.0	15.0	605	1”	B600	GG455	2	2	full	404.4	S20220406S0240
BA2	5 S	1200	⌖	0.0	15.0	605	1”	B600	GG455	2	2	full	490.9	S20220406S0241
	✓ TCS ✓ GCAL ✓ GMOS	15:32 (932 s) Remaining												
	6 F			0.0	15.0	605	1”	B600	GG455	2	2	full		

⚠ **This observation should be interrupted due to degrading conditions.**
Complete step 6 then switch to 22A-Q-301 [27] in the new plan.

Deactivate non-essential steps | Switch to 22A-Q-301 [27]

	Step	Exp (sec)		p	q	λ (nm)	FPU	Disperser	Filter	Xbin	YBin	ROI	S/N	Dataset(s)
AA1	7 F			0.0	0.0	600	1”	B600	GG455	2	2	full		
	8 S	1200	⌖	0.0	0.0	600	1”	B600	GG455	2	2	full	564.2	
	9 S	1200	⌖	0.0	0.0	600	1”	B600	GG455	2	2	full	636.0	

22A-Q-222 [93] Science: 5/14 Steps (1:05:32 Remaining, 0:05:32 in Scheduling Unit)

Figure 16: Observe mockup showing current and suggested plan with plan change guidance.

Plan change guidance would cover Target of Opportunity switches as well, and is a requirement for supporting GEMMA.

6 Implementation, Testing, and Deployment

Here we detail the technology choices, our testing strategy, and the deployment details for the Scheduler.

6.1 Implementation Details and Third-Party Frameworks

The Scheduler is implemented in Python 3.9 and provides GraphQL APIs. Python was selected as the development language of choice for the Scheduler project because both the developers and science staff are familiar with writing Python code and can thus contribute to the project.

In addition, Python is widely used in the astronomy community and thus offers many libraries that are useful, high in efficiency, and have been engineered to work together for mathematical, scientific, and astronomical computation, which we leverage to perform calculations that provide the Scheduler with the information necessary for it to produce plans. The project uses NumPy to increase the speed of calculations wherever possible, and AstroPy to provide common core functionality for astronomy. Additionally, the AstroPy library assists in development by making sure that values used in computations are in the required units, and that units are used sensibly together.

One risk to the use of AstroPy is that it performs many calculations that are unnecessary to the Scheduler, which may down plan generation performance. In order to increase the efficiency in generating a plan, once we have confidence that the Scheduler is producing sane, valid plans, we intend to attempt to remove AstroPy from the code base and instead reply solely on NumPy and native Python structures for measurements and quantities like angles, distances, dates, and times.

The GPP project relies on GraphQL to disseminate and consume information from other services. GraphQL is particularly efficient because GraphQL interfaces specify precisely the information that they require, and GraphQL supports a subscription-based model where a subscriber to a service will receive a notification when existing data of interest is mutated or new data becomes available. Python offers several implementations of both GraphQL clients and servers, and the Scheduler will need to act in both roles.

6.2 Testing and Deployment

There are two types of testing that are performed on the Scheduler, depending on the feature under test. These are:

- Specifications-based testing, used when possible to do so, which consists of testing the functionality of parts of the Scheduler against requirements. A specification-based testing approach is preferred when feasible, since the specification testing engine is provided with information on how to construct the test data, and then performs many tests to ensure that the requirements are satisfied by the results. If a specification test fails, the engine attempts to reduce the complexity of the data that caused the failure to occur to generate the simplest counterexample violating the requirements. Another advantage to specification testing is that from the instructions on how to construct the test data, the testing engine attempts to determine edge cases. For specifications-based testing, we have chosen to use the Hypothesis library.

- When specification-based testing is not possible, we use unit testing to ensure that specific calculations done in the Scheduler result in known, expected values. These tests are implemented using the pytest framework.

The Scheduler source code is stored in GitHub, and configured to run via GitHub Actions. When a pull request (i.e. a request to merge changes to the codebase) is submitted, the test cases are run against the repository with the changes in place, and if any of the test cases fail, GitHub disallows the pull request to be merged. Another team member must additionally review the code changes as a sanity check to detect any possible issues.

We are using Heroku as our cloud platform, as it offers services that simplify management compared to raw Amazon Web Services. It allows automated application updates, offers on-demand resizing and scaling with integrated load-balancing, and has tight integration with GitHub for testing and deployment. This is the same platform that GPP is using.

Appendix A - Timing Constraints

Each observation can have timing constraints that define the time windows, or intervals, when an observation can be executed. The timing windows are a list of the form [{start(t) duration(dt) repeat(n) period(p)}, {start(t) duration(dt) repeat(n) period(p)}, ...]

```
t
|==========|------|==========|---
     dt
\-----------------\
         p - repeat n times
```

where:

- `duration(dt) = -1`, open ended
- `repeat(n) = -1`, repeat forever

For inter-observation constraints, we need:

- `Idprev` = id of previous observation/visit
- `Idnext` = id of next observation/visit
- `Delay(d)` = delay after previous observation ends, 0 if consecutive
- `Duration(w)` = duration of window, -1 if open ended

A.1 Custom cadence

Each observation can have an 'internal' set of timing windows that is used by the Scheduler. This combines all of the various timing constraints (e.g. user defined timing windows, inter-observation constraints, program start and end dates); therefore, all observations have internal timing windows, even if none were defined by the user. Visits inherit the timing windows of the observations that they contain.

The observation timing windows and the inter-observation constraints work together. If visit1 and visit2 have the same set of repeating timing windows of an eclipsing binary, but visit2 must be taken between `d` and `d+w` hours after visit1, then visit2 can be scheduled during the timing windows that are in that interval. If there are none, then a warning should be given.

visit1 - length 6
```
t1   dt1
|==========|------------|==========|------------|==========|---
```
Idnext = 2

visit2 - length 6
```
t2   dt2
|==========|------------|==========|------------|==========|---
```

```
Idprev = 1, d = 34, w = 12
```

Note that visit1 is scheduled in the first timing window (represented by the `*` symbols below). Therefore, visit2 can be scheduled between `d` and `d+w` after the end of visit1 (within the `//` below). This overlaps with one of visit2's timing windows, denoted by `|===`. Note that there could be cases in which none of the timing windows fall in the interval `[d:d+w]`. If this happens, then a warning should be given. Let's say that the scheduler puts visit2 in the interval that is available as denoted by the + symbols. The plan then looks like the timeline below.

```
|===******=|                                   visit2
    Visit1---------------------------------/~~~|++++++==/
                          d                        w
```

GreedyMax will handle these by scoring visit$_1$ and visit2 independently. Until visit1 is scheduled, visit2 must not be considered, so it is given no timing windows or alternatively a score of 0.
In this case, visit1 has to be done before visit2. If visit1 is placed in the plan, then GreedyMax updates the score and timing windows for visit2.

Appendix B - Scheduler Data Model

We present the data model as used internally by the Scheduler. This represents the GPP science program as shown in Figure 1

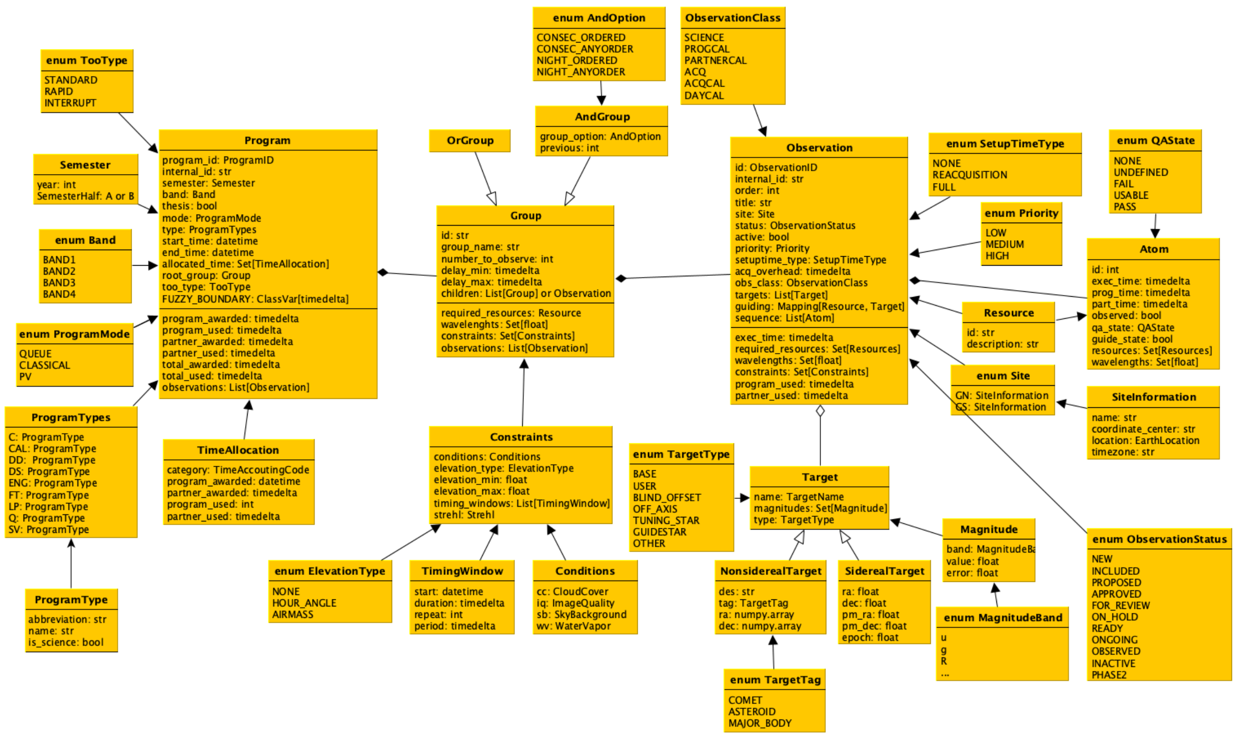


Figure 17: The representation of a science program in the Scheduler.

Program is the top class in the science program hierarchy which contains information relevant across the whole Program, e.g. the semester during which the Program has been accepted, the start and end dates of the Program, whether the Program is for a thesis, and whether the Program represents a Target of Opportunity. The *FUZZY_BOUNDARY* is a class constant that allows for a fuzzy boundary for a program's start and end times.

Each Program contains a set of **TimeAllocation** objects. These maintain information on a partner-by-partner basis, identified by the **TimeAccountingCode,** regarding the time awarded to a program and the time that has been used thus far, divided between program time and partner calibration time. While we do not display the TimeAccountingCode in Figure 17, it is simply an enumeration for identification purposes of the different partners that contribute to Gemini or the various Gemini projects.

An **Observation** represents **Resource** configuration data and information identifying when a sequence of steps - called **Atoms** - can be performed at a given **Site**, which includes information about its location, time zone, and coordinate center (used for HORIZONS lookups). An Observation contains a **Target** of interest, which can be either a **SiderealTarget** or a **NonSiderealTarget**, along with other target environment and guiding information. The **Atoms** comprise the steps necessary to complete the Observation: each Atom is a collection of steps that is the smallest unit of execution that yields useful data.

Observations have **Constraints** that represent the **Conditions** such as the **SkyBackground** or level of **CloudCover** (see Section 3.2), which are represented by bucketed enumerated values, e.g SB50 and CC70 respectively, and other information regarding elevation constraints, determined by the **ElevationType** - either airmass or hour angle - and the respective elevation requirements.

Observations are organized into **Groups.** The Group class is an abstract class that represents the concepts of the various scheduling groups as mentioned in Section 2.2. The concrete implementations of a Group are either **AndGroup** or **OrGroup**, which can contain nested subgroups and Observations and represent what constitutes completion for a Program. A Group can roll up its contents - Observations and their respective Atoms - to determine the collection of Resources, Constraints, Wavelengths, and Sites used by the Observations contained in the tree rooted at that Group.

An AndGroup requires all of its subgroups and Observations to be performed to be considered completed, and has parameters that allow for specific time constraints to be imposed on what order and when - either globally or relative to each other - the Group’s Observations must be performed. Along with **TimingWindow** information, which determines when an Observation can be performed, the Scheduler allows for complex relationships such as cadences to be modeled as described in Section 2.2 and Appendix A.

An OrGroup specifies how many of its subgroups or Observations must be executed for the OrGroup to be completed.

In the data model itself, the Scheduler has no built-in restrictions on how Groups can be nested and allow for arbitrary nesting: all restrictions are implemented in the API that populates the Program structure or in the Selector when trying to calculate scoring and time slot availability.

In order to determine:

- the Resources necessary for the Program or subgroup
- the most restrictive Conditions required by the Program or subgroup
- timing information for a Program or subgroup

the Scheduler simply traverses the relevant section of the tree. Every Program has a Root Group node (described in Section 2.1) from which the Scheduler can traverse the tree to calculate Program-level information.

Each Group has information contained in it that is used to determine when the Group (and the subgroups and Observations it contains) can be executed per night and per time slot over the night. The information is assembled by the **Collector** (see Section 4.2.1), which also calculates information per night per site regarding night events, which are stored in the **NightEvents** object, and then uses this information to determine visibility calculations for the Target of each Observation, which is stored in the **TargetInfo** object.

After this information has been calculated, the **Selector** (see Section 4.2.2) iterates over the groups for each Program that has been determined to be a candidate for the plan, and conglomerates the data into a **GroupInfo** object which contains information about when a Group can be scheduled (i.e. for a given night at a given site, the time slots during which it can be executed), and assigns each Group a score per time slot, which determines the relative value of scheduling the Group during that time slot for the plan.

Both **TargetInfo** and **GroupInfo** are calculated by the Scheduler and are not direct inputs.

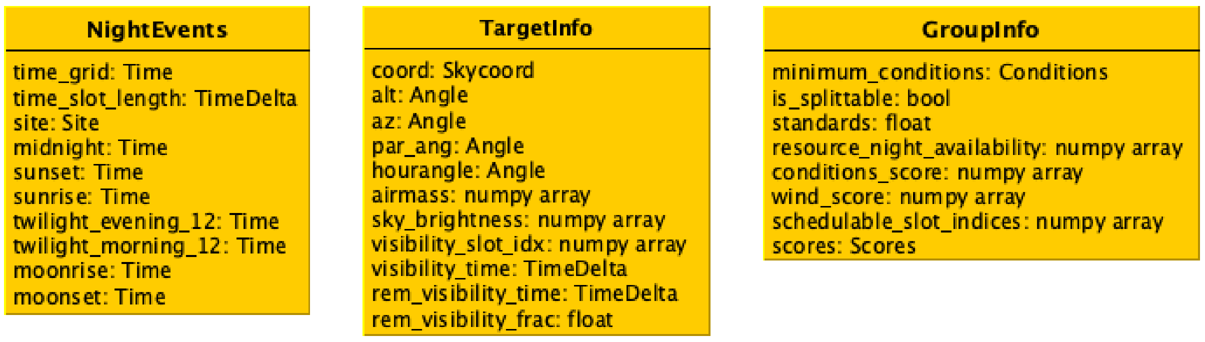

NightEvents
time_grid: Time
time_slot_length: TimeDelta
site: Site
midnight: Time
sunset: Time
sunrise: Time
twilight_evening_12: Time
twilight_morning_12: Time
moonrise: Time
moonset: Time
TargetInfo
coord: Skycoord
alt: Angle
az: Angle
par_ang: Angle
hourangle: Angle
airmass: numpy array
sky_brightness: numpy array
visibility_slot_idx: numpy array
visibility_time: TimeDelta
rem_visibility_time: TimeDelta
rem_visibility_frac: float
GroupInfo
minimum_conditions: Conditions
is_splittable: bool
standards: float
resource_night_availability: numpy array
conditions_score: numpy array
wind_score: numpy array
schedulable_slot_indices: numpy array
scores: Scores

Figure 18: The classes containing computed data about night events, target information, and group information as per the Scheduler components.

Appendix C - Schedule Mockups

The following are sketches of how the three different interfaces for Schedule might look, demonstrating the various features we believe they should contain. They were used in large part to help determine what features a user would need to see or have. These are **not** expected to be part of a final design but as a guide for what might be needed in a final design.

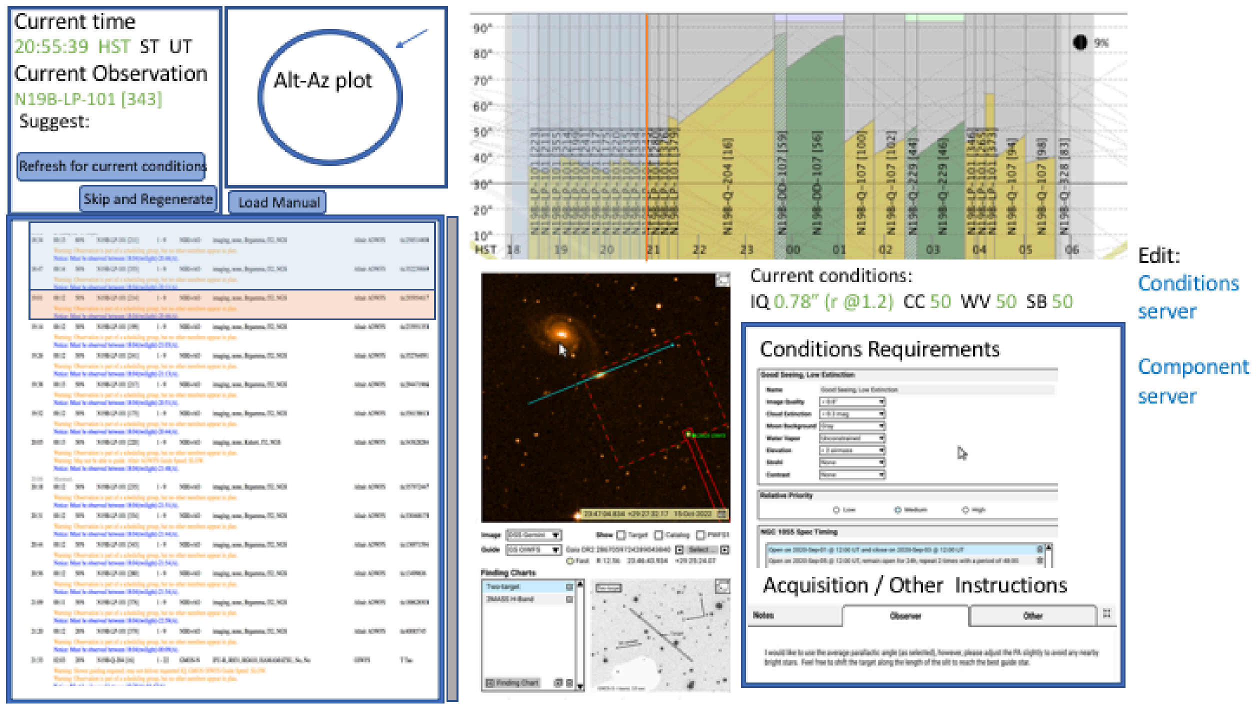


Figure 19a: Sketch of possible layout of Schedule GUI.

This includes some components that would normally have come from the OT. Explore can be used by the observer, so this may not need to include everything that is found instead in Explore. The critical components for the Scheduler nightly plan include the plan for the current conditions, information about each observation, highlight of the current observation, conditions information, a way to skip an observation, an option to load a manually created plan, Alt-Az and El-Time plots, and quick links to the Conditions and Components servers. If loading a Manual plan, the options will include stored plans along with the option to revert to Auto. Observe indicates when you are in Manual vs. Auto mode. The Refresh for current conditions button will be used when conditions change, as per Figure 19b. It is possible to specify conditions for observing as well, e.g. in the event that clouds are coming in but have not yet arrived. This does not currently allow for limits to the dome alt-az to be applied, which might be better suited for the Component server.

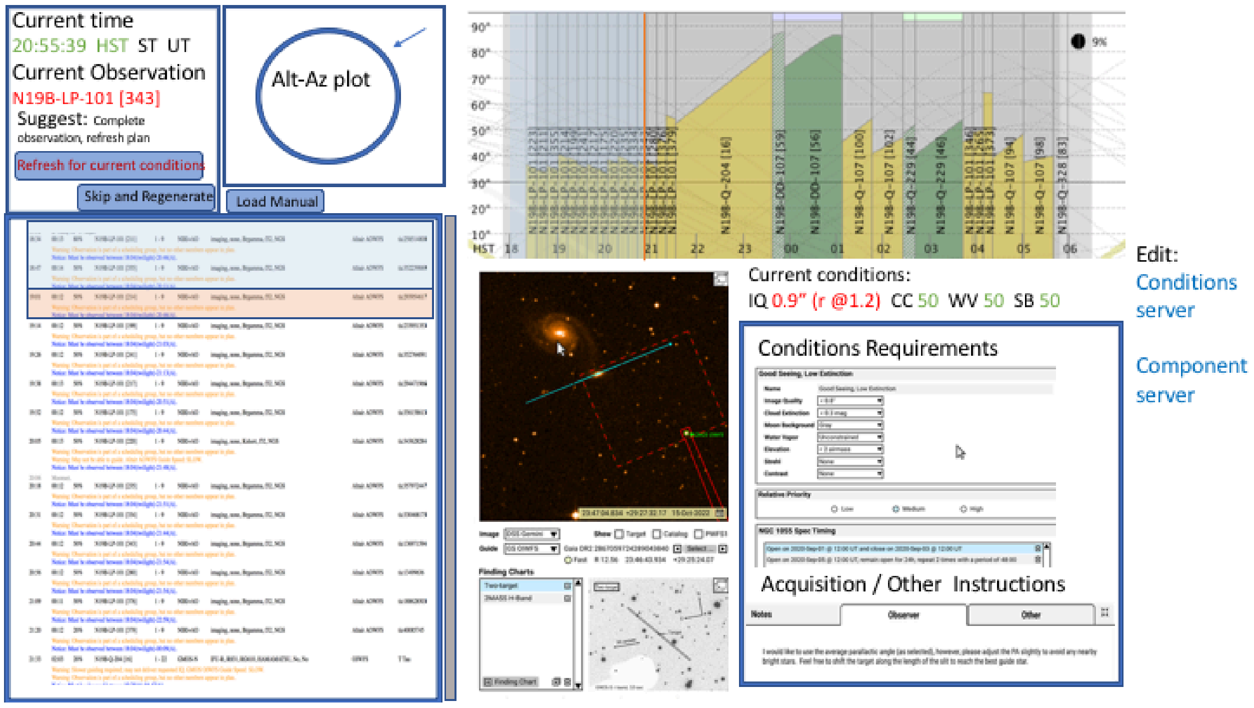


Figure 19b: Deterioration in conditions.

In the event that conditions deteriorate and the current observation goes out of spec, current conditions and current observation are displayed in red and accompanied by a warning or pop-up. The Scheduler provides a suggestion to the observer about what to do (continue, stop, abort, continue for two more steps to complete the pair, etc).

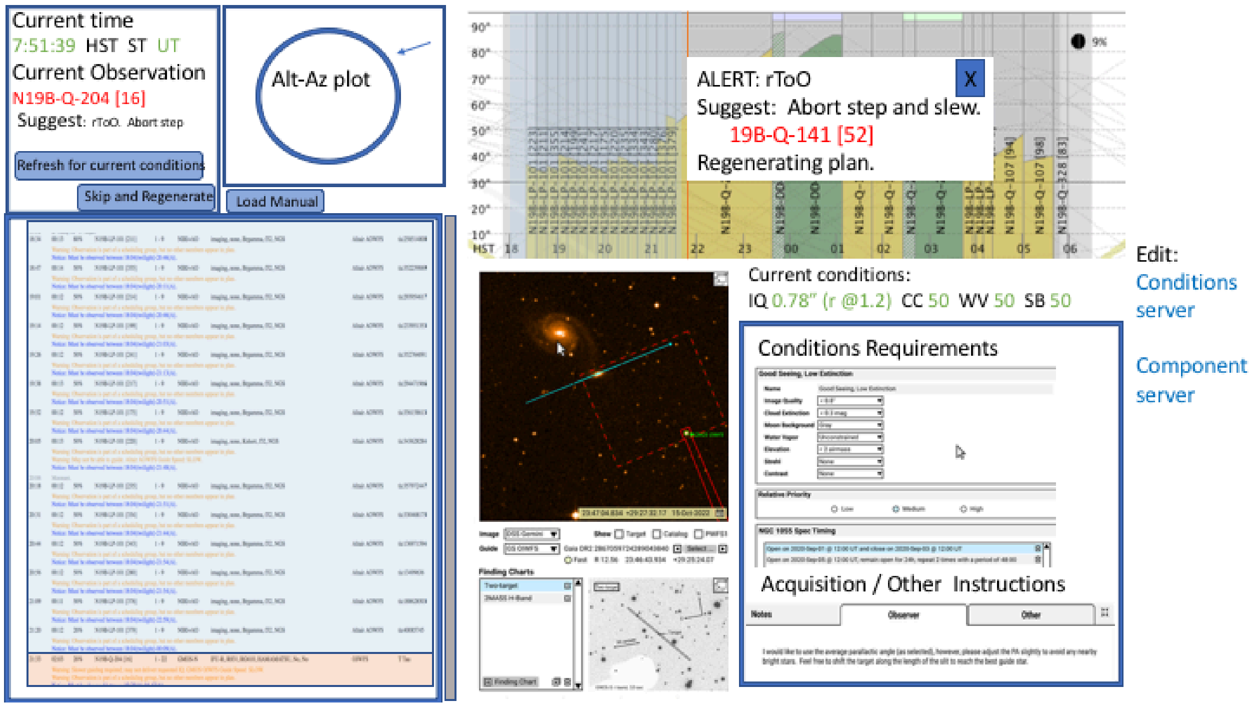
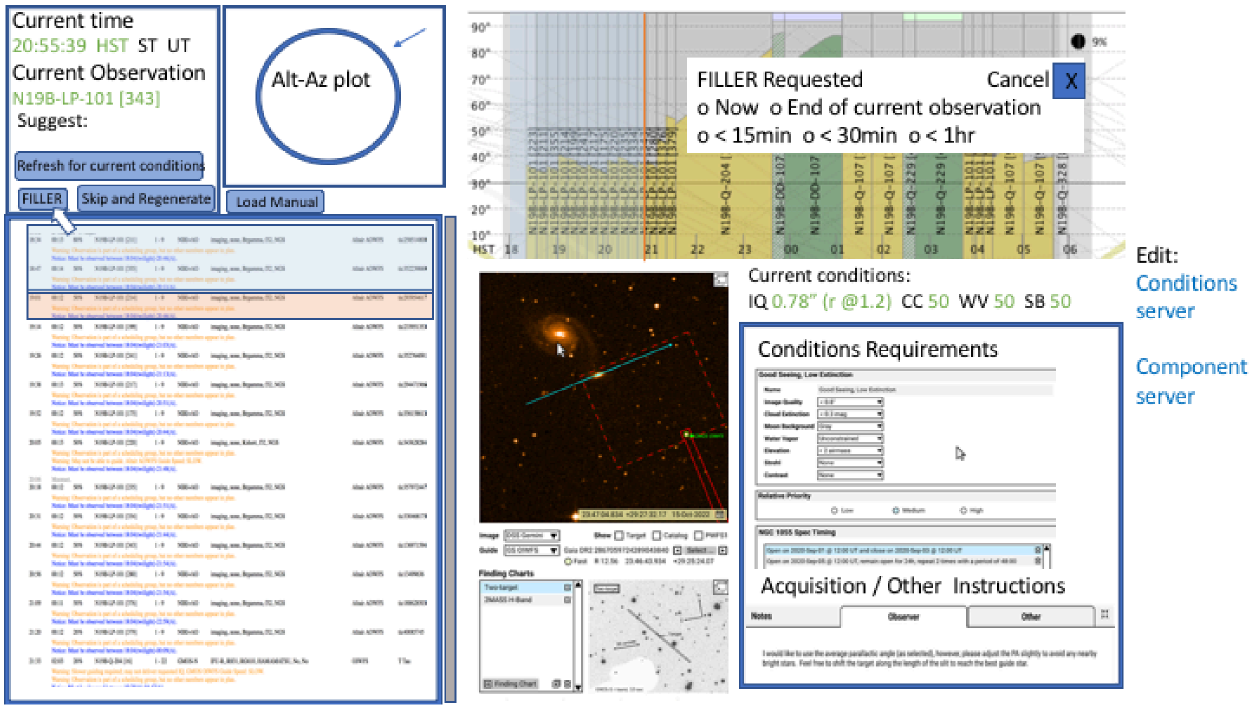


Figure 19c: Rapid ToO event.

In the event of an rToO, the current observation is displayed in red, an alarm sounds, and a pop up announces the ToO with a suggestion of what to do next. These suggestions will include options to stop, finish the current step and stop, abort, etc. Upon acknowledging the pop-up, the Scheduler regenerates the plan with the rToO included. The suggestion remains under the Current Observation until the next slew, or until no longer relevant.


Current time
20:55:39 HST ST UT
Current Observation
N19B-LP-101 [343]
Suggest:

FILLER Requested Cancel X
o Now o End of current observation
o < 15min o < 30min o < 1hr

Current conditions:
IQ 0.78" (r @1.2) CC 50 WV 50 SB 50

Figure 19d: The FILLER button.

This can be used in cases such as where the observer is aware that some Eng work will start as soon as personnel arrive and needs the Scheduler to supply short observations, or if the night crew is troubleshooting a fault and expect to be able to use the previous instrument / setup again shortly, or if a PV observer has a hole in their plan, etc. It might also be useful any time the observer can input a specific length of time (rather than using the quantized options shown here).

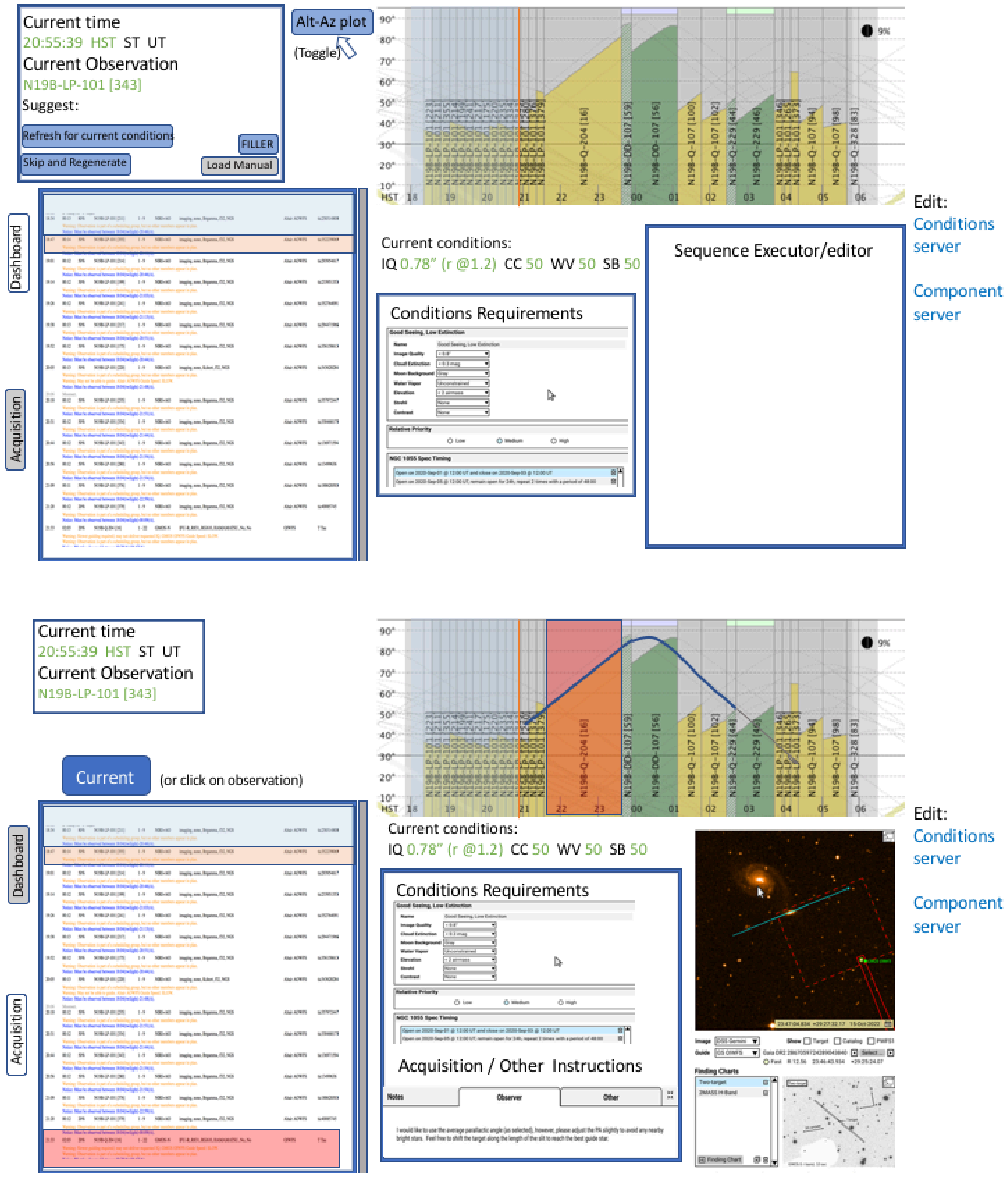
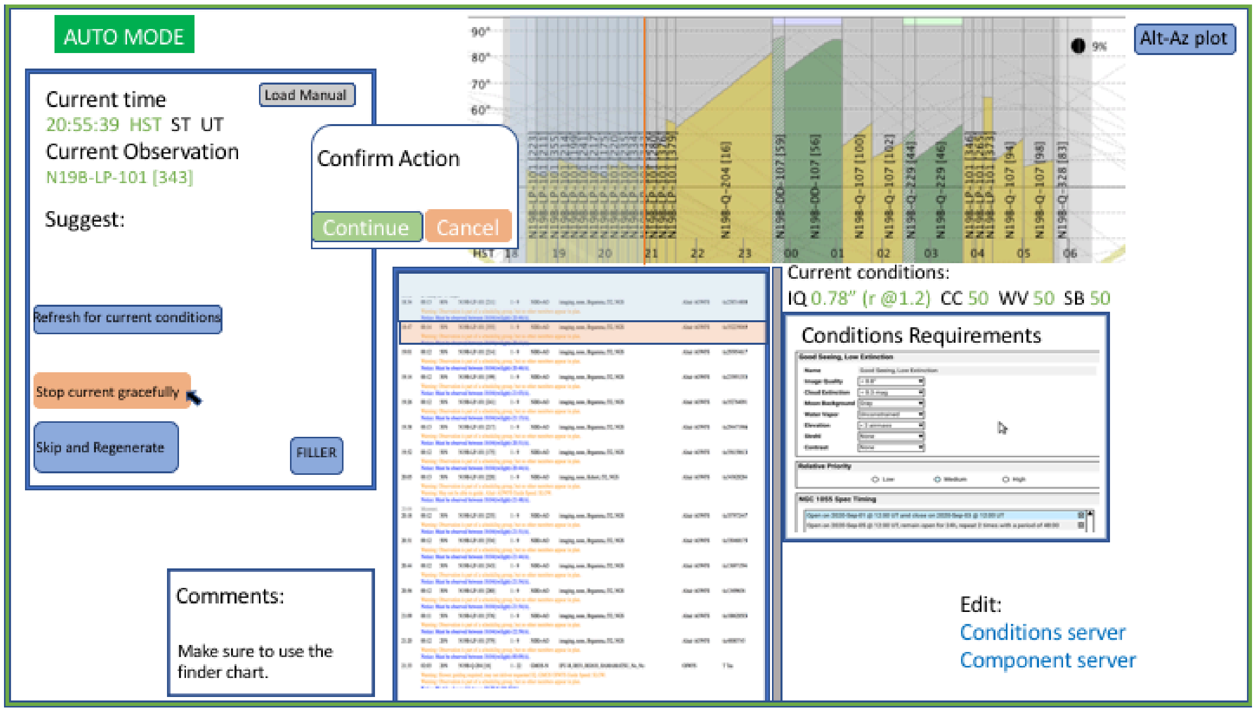


Figure 19e: Tabs on the left reduce crowding in the interface.

The option shown here has two tabs: one for acquisition, which includes finder charts, position editor, and guide star setup; and one for the dashboard to display while otherwise observing, which includes the sequence executor as well as information about the current and upcoming observations and current and required conditions.



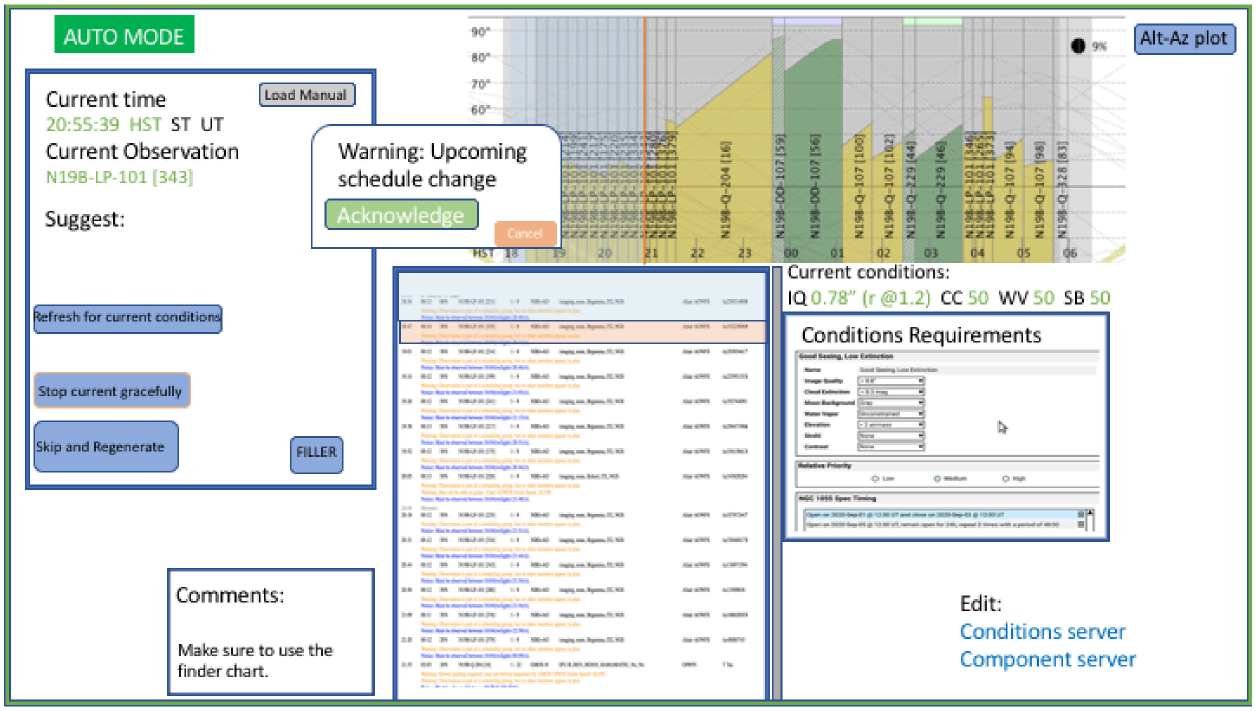
Current time
20:55:39 HST ST UT
Current Observation
N19B-LP-101 [343]
Suggest:

Current conditions:
IQ 0.78" (r @1.2) CC 50 WV 50 SB 50

Comments:
Make sure to use the finder chart.

Figure 19f: Example of only including the necessary Scheduler components.

There is a "Stop current gracefully" button to allow the observer to request for a stop of the current observation after the current atom is completed (including the minimum number of steps and required calibrations). A popup requests confirmation of the action, which would similarly be needed for the other button options. Other buttons include "Refresh for current conditions" for when weather changes occur; "Skip and Regenerate" for cases where the observer cannot proceed with the current suggestion (e.g. finding chart unclear); and "Filler" for when a short filler is desired. The AUTO MODE indicator at the top informs the user that this is Auto and not Manual mode. A different color is used to denote MANUAL MODE. When in Manual mode, the "Load Manual" button is replaced with a "Load Auto" button. The elevation plot can be swapped to alt-az with the toggle button. A comments box displays any important information for the observer about the current observation. Current and requested conditions are displayed. The observer is able to click on an upcoming observation, highlighting it, and the conditions / comments are displayed for that observation. All components showing information about the upcoming rather than current observation are colored / shaded, indicating that information is not currently relevant.



Current time
20:55:39 HST ST UT
Current Observation
N19B-LP-101 [343]

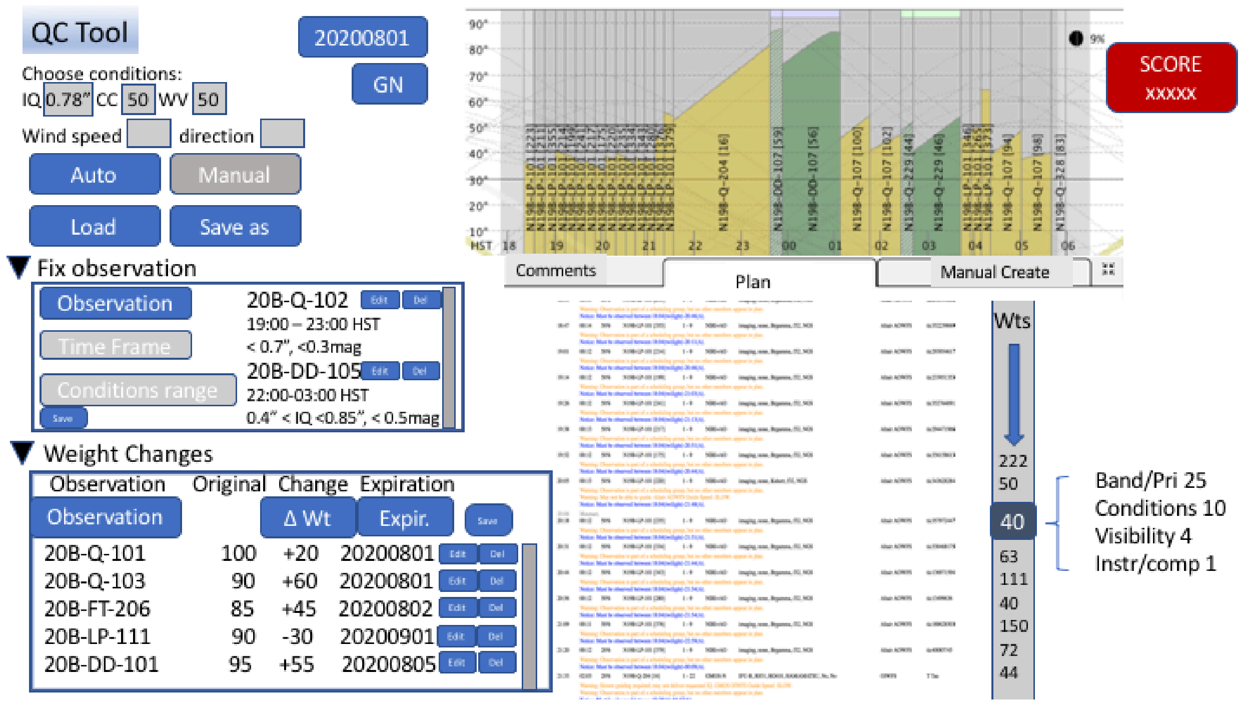
Current conditions:
IQ 0.78" (r @1.2) CC 50 WV 50 SB 50

Comments:

Make sure to use the finder chart.

Figure 19g: Notification that the Scheduler is going to make changes to the plan that doesn't require anything from the observer (i.e. the change will take place after the current observation).

This will be in the form of a pop-up like the rToO and FILLER ones, but also with an audible bell to inform the observer that the schedule has changed and to acknowledge by closing the popup: in this case, we may want the observer to have a say in whether the change is made or just acknowledge that they know this change occurred. These changes can occur for any changes in the database, in conditions, or instrument / component availability. This is to avoid the case where the plan should just change and the observer runs the risk of not noticing.



Choose conditions:
IQ 0.78" CC 50 WV 50
Wind speed direction

▼ Fix observation

Observation	20B-Q-102 Edit Del
	19:00 – 23:00 HST
Time Frame	< 0.7", <0.3mag
	20B-DD-105 Edit Del
Conditions range	22:00-03:00 HST
Save	0.4" < IQ <0.85", < 0.5mag

▼ Weight Changes

Observation	Original	Change	Expiration		
Observation		Δ Wt	Expir.	Save	
20B-Q-101	100	+20	20200801	Edit	Del
20B-Q-103	90	+60	20200801	Edit	Del
20B-FT-206	85	+45	20200802	Edit	Del
20B-LP-111	90	-30	20200901	Edit	Del
20B-DD-101	95	+55	20200805	Edit	Del

Wts: 222, 50, 40, 63, 111, 40, 150, 72, 44

40: Band/Pri 25, Conditions 10, Visibility 4, Instr/comp 1

Figure 20: The daily QC GUI, also known as the Schedule application.

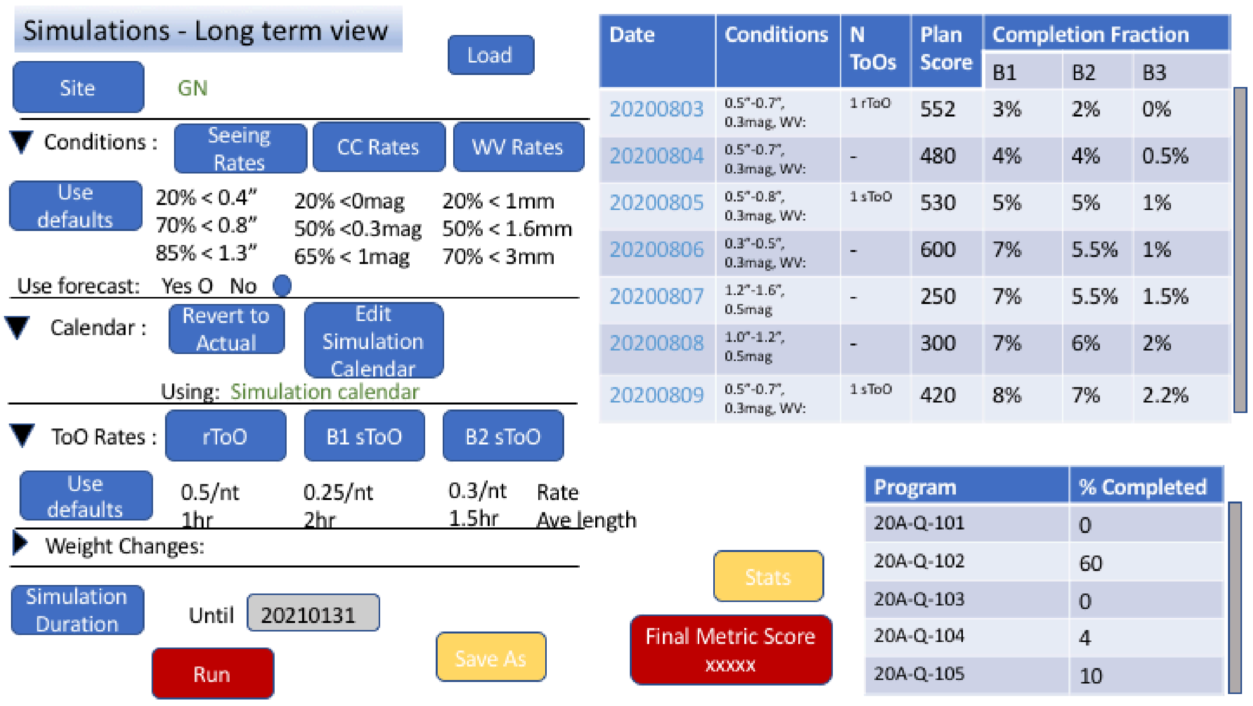
With the Schedule application, the QC can choose expected conditions for the night (along with the date and site) and create an automatic or manual plan for the upcoming night. The plan shows up in the Plan tab, including the weights for individual observations along with information about how those weights were derived. The total score for the plan is also shown. The weights and score are provided for manual plans as well. The interface allows a QC to force an observation to be included in an automatically generated plan using the Fix Observation option. The QC chooses an observation and applies a time frame and range of conditions for including in the plan. These observations will be displayed and can be edited or deleted. The QC is also able to make weight changes to observations by applying ‘bumps’ or delta weights. These modified weights will have an expiration date, but it will be possible to edit or delete these: they will be displayed until the changes are removed. Not shown here is a way for QCs to be able to adjust weights for all observations using the same systems and components (e.g. NIRI or R831). Changes from above include wind speed and direction options at the top. Modification options are expandable.

Using the Manual Create tab, the QC can create a plan in a similar way as is done with the QPT for the conditions listed in the top left, and can save this plan, which can be later loaded here or in the Observe dashboard. The scores for each candidate will be provided for all available observations. If the QPT is maintained to work with GPP, it could be used instead to generate a plan for a specific set of conditions and then saved and loaded into the Schedule view or into Observe.

Not included in this mockup is:

- a list of observations that are no longer schedulable in the semester for a given target, setup, and/or conditions;
- a list of programs that can no longer reach >= 80% completion with the currently defined observations; and
- a list or quick access to a list of unfinished programs that are about to or have expired within the past week.

In these three cases, the QC may need to contact the PIs to make adjustments to their programs so they will have quick access to this information through this tool.



Simulations - Long term view

Load

Site: GN

Conditions : Seeing Rates | CC Rates | WV Rates

Use defaults

Seeing Rates	CC Rates	WV Rates
20% < 0.4”	20% <0mag	20% < 1mm
70% < 0.8”	50% <0.3mag	50% < 1.6mm
85% < 1.3”	65% < 1mag	70% < 3mm

Use forecast: Yes O No ●

Calendar : Revert to Actual | Edit Simulation Calendar

Using: Simulation calendar

ToO Rates : rToO | B1 sToO | B2 sToO

Use defaults

rToO	B1 sToO	B2 sToO	
0.5/nt	0.25/nt	0.3/nt	Rate
1hr	2hr	1.5hr	Ave length

Weight Changes:

Simulation Duration: Until 20210131

Run

Save As

Date	Conditions	N ToOs	Plan Score	Completion Fraction		
				B1	B2	B3
20200803	0.5"-0.7", 0.3mag, WV:	1 rToO	552	3%	2%	0%
20200804	0.5"-0.7", 0.3mag, WV:	-	480	4%	4%	0.5%
20200805	0.5"-0.8", 0.3mag, WV:	1 sToO	530	5%	5%	1%
20200806	0.3"-0.5", 0.3mag, WV:	-	600	7%	5.5%	1%
20200807	1.2"-1.6", 0.5mag	-	250	7%	5.5%	1.5%
20200808	1.0"-1.2", 0.5mag	-	300	7%	6%	2%
20200809	0.5"-0.7", 0.3mag, WV:	1 sToO	420	8%	7%	2.2%

Stats

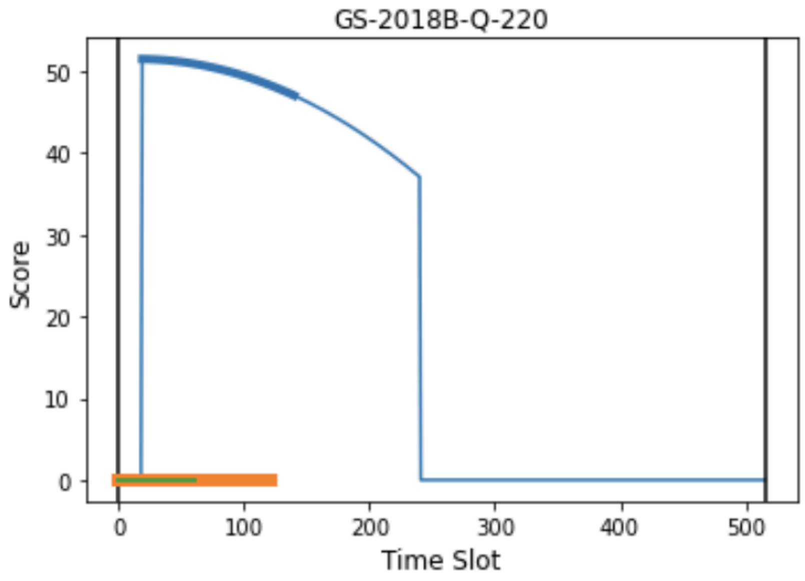
Final Metric Score xxxxx

Program	% Completed
20A-Q-101	0
20A-Q-102	60
20A-Q-103	0
20A-Q-104	4
20A-Q-105	10

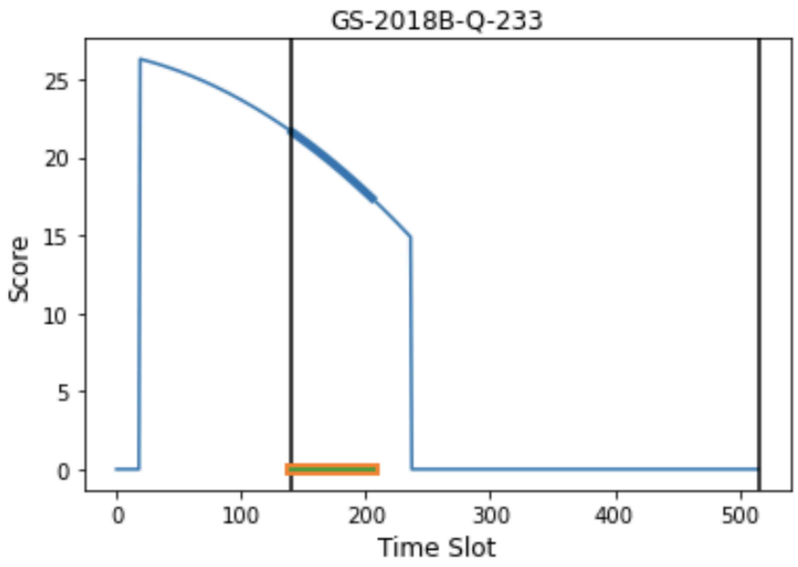
Figure 21: The GUI for longer term planning and simulation testing.

In the Simulation GUI, the user sets the site; conditions; rate of r/sToOs and average ToO observation length, using mean and sigma; and simulation duration. Default settings for Conditions and ToO rates can also be used. The user also chooses between the actual semester telescope / instrument calendar or a simulated one in which they can make edits. The simulation calendar starts by default with the semester calendar information, and it is possible to store edits to the simulation calendar as files that can be selected for later use. All modification options in this interface as shown here are expandable. The simulator makes use of the Resource service personnel calendar in order to know when to allow instrument component changes, as the Simulator will have to be able to make decisions about when to implement component changes. After running the simulation, the dates for that period are displayed in a table along with the simulated conditions and number of ToOs for each night, the total plan score for the night, and running completion rates for the different bands. Another table lists each active program in the database and the completion achieved by the end of the simulation. The total metric score for the period given this completion is also displayed. Each simulation can be saved and reloaded to view it. One of the Operations aims is to see other statistics (e.g. completion rates) for each instrument, currently shown as a Stats button. It is also possible to change the weights of the observations, in the same way as for the real plan, but which would be applied only in the simulations. Actual weather forecasts can be used, either via reading these in digitally or by allowing input for upcoming dates. Additionally, the plans generated for each night can be viewed as links in the dates in the top table as shown here.

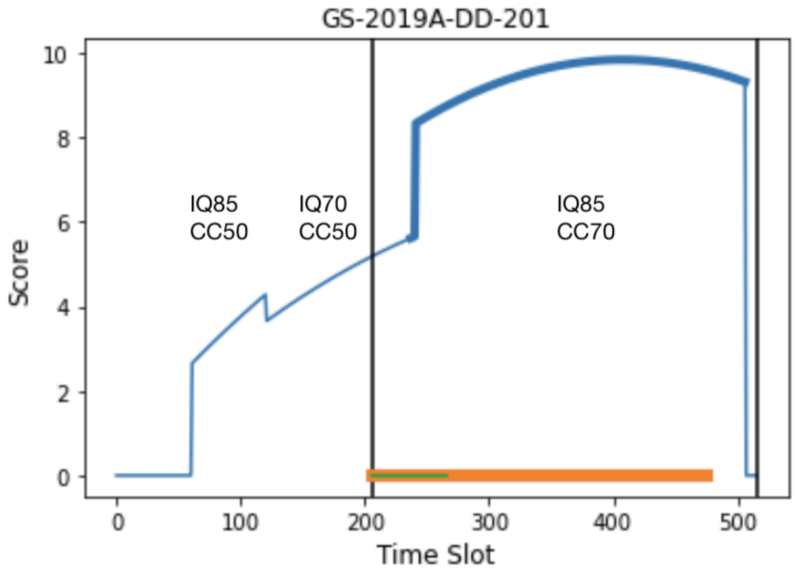
Appendix D - GreedyMax Example



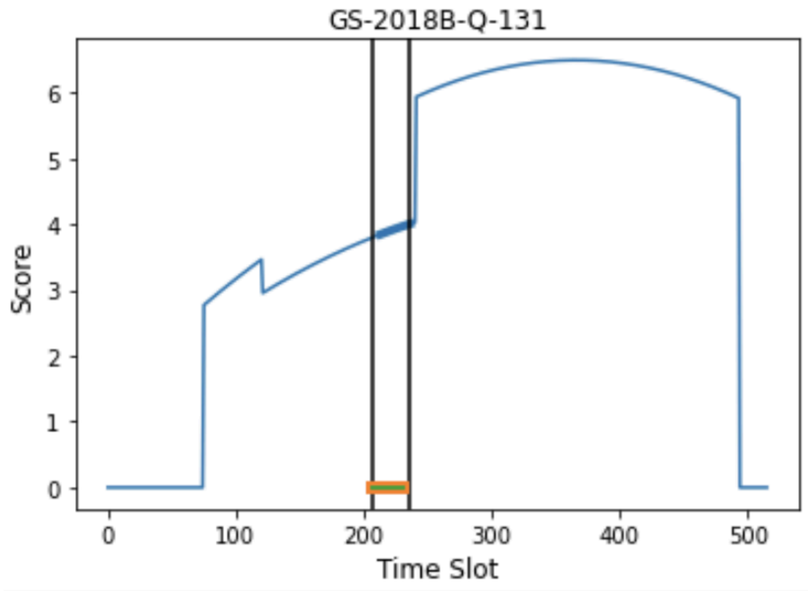
This gives an example of the creation of a queue plan using the GreedyMax Optimizer. These vertical black lines show the time interval that the algorithm is considering at that iteration. Each figure shows the visit score versus the time slot for the visit with the maximum score in the time interval. The thinner blue line shows when the score is non-zero. This is constrained by airmass, sky brightness, cloud cover, image quality constraints, timing windows, and wind speed if it is above the limit. The thick orange line represents the full length of the visit and the thinner green line represents the minimum visit length. The thicker blue section of the curve is where the visit has been scheduled. In the case above, GreedyMax maximized the integral of the score over the observation length to position it as early as possible in the time interval.



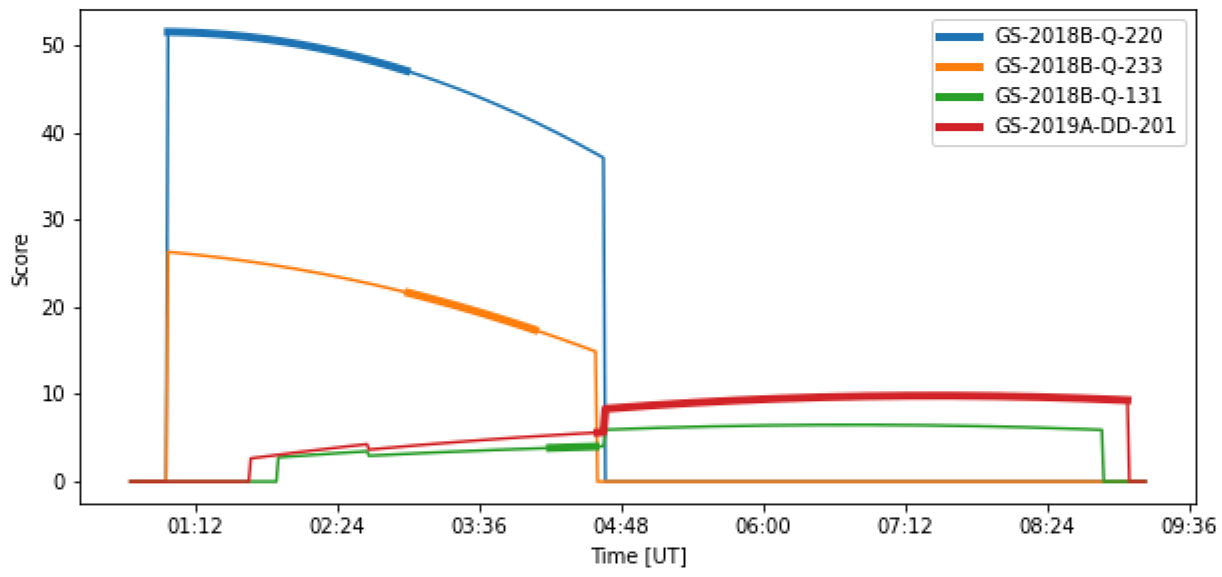
The next iteration is similar to the first: the integral of the score has been used to position the observation. In both of these cases, the downward shape of the scores comes from the hour angle term. These targets are setting early in the night and GreedyMax is scheduling them as early as possible.

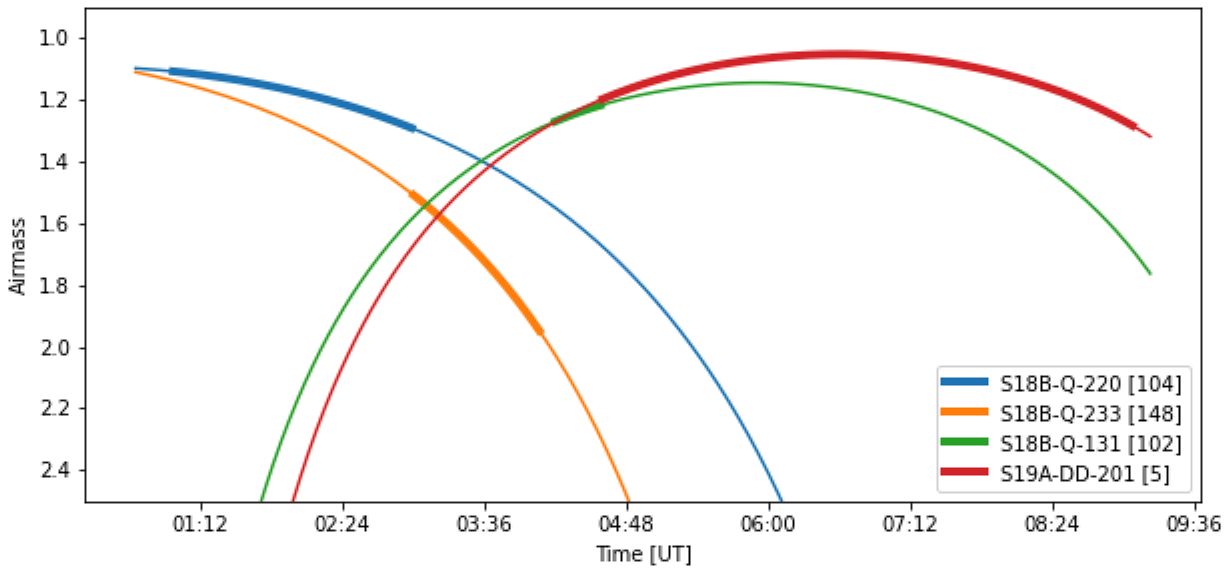


The third iteration shows the impact of using a forecast. The actual conditions can be time-dependent. Early in the night the conditions are IQ85/CC50. Then they improved to IQ70/CC50 before some thin clouds and poor IQ arrived (IQ85/CC70). The selected visit has conditions constraints of IQ85/CC50. The dip in the score function around time slot 200 shows how the score is reduced when the conditions are better than required.



Lastly, the plan is finished by inserting a short visit into the remaining schedulable interval. The final plan, plotted using both visit score and airmass, is given below.


Score
50
40
30
20
10
0
01:12
02:24
03:36
04:48
06:00
07:12
08:24
09:36
Time [UT]
GS-2018B-Q-220
GS-2018B-Q-233
GS-2018B-Q-131
GS-2019A-DD-201


Airmass
1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
01:12
02:24
03:36
04:48
06:00
07:12
08:24
09:36
Time [UT]
S18B-Q-220 [104]
S18B-Q-233 [148]
S18B-Q-131 [102]
S19A-DD-201 [5]