RTS2: meta-queues scheduling and its realisation for FLWO 1.2m telescope

Petr Kubánek[a], Emilio Falco[b], Martin Jelínek[c], Michael Prouza[a], Jan Štrobl[d], Martin Fuchs[e] and Javier Gorosabel[c]

[a]Institute of Physics, Academy of Sciences, Praha, Czech Republic;
[b]SAO/FLWO, Amado, AZ, USA;
[c]Instituto de Astrofísica de Andalucía CSIC, Granada, Spain;
[d]Astronomical Institute, Academy of Sciences, Ondřejov, Czech Republic;
[e]Štefánik Observatory, Petřín, Praha, Czech Republic;

ABSTRACT

RTS2, or Remote Telescope System 2nd Version, is an open-source, distributed and modular observatory control system. During the course of its development lasting over a decade, the original goal to develop software capable of searches for optical transients of γ-ray bursts changed to develop a system for full control of large observatories executing complex observing scenarios.

In this presentation, we would like to share our experience with meta-queues scheduling, developed primarily for automation of the FLWO 1.2m telescope. Meta-queues scheduling allows observers to quickly build and combine different observational scenarios, while still retaining ToO and weather interruption capabilities. Thanks to the queues and scheduling graphical user interface, observers can use the system without the need to understand complex functions used in the traditional merit function scheduling. By combining meta-queues and merit function scheduling, observatories can offer different options to schedule their observations to their users, so the acquired data will match the observers' expectations.

Keywords: autonomous observatory, robotic telescope, telescope network, RTS2

1. INTRODUCTION

The development of software which operates an autonomous observatory is not an easy task. The development of software which allows observatory users with different scheduling requirements to schedule observations, obtain the observations, process the observations and perform other tasks associated with running an observatory is a task at least an order of magnitude harder.

It is of course not easy to design and code scheduling for a space-based observatory,[1] but it is even more challenging to design and code scheduling for a ground-based observatory. The ground environment is more dynamic and less predictable than the space environment. Scheduling should be able to adjust night plans based on changing atmospheric conditions. It should allow observatory users to re-arrange observed objects so the observations they would like to get done while sleeping would be done in proper order.

Mid-size and large ground based observatories usually serve various user groups with various scheduling needs. Software which enables users to observe a single star for most of the night might be ideal for exoplanet researchers, but does not fit the needs of supernova observers.

Scheduling of observing time is a major factor contributing to the success of any autonomous observatory. Autonomous observatories with perfect scheduling succeed, whereas those without it fail. Multiple attempts were made throughout history to solve this problem – for example, see Stella[2] or Robonet[3] scheduling. Most of these rely on some kind of merit function, which evaluates target "usability". The scheduler always picks the target with the highest merit function value. Brief introduction to merit function scheduling is provided in section 2.

Further author information: (Send correspondence to P.K.)
P.K.: E-mail: kubanek@fzu.cz, Telephone: +420 737 500 268; Project web site: `http://rts2.org`

Although various systems based on merit-function scheduling may offer a programmer a perfect plaything, they are very difficult to explain to the investigators. Astronomers are not interested in hearing how they can make their observations by adjusting knobs which influence merit function. They would like to have their targets observed using a simple and elegant method to do so. That is where queues, used at most of the large observatories[4],[5] come into action. Observers can place their targets into a queue and the queue is executed. For detailed discussion of queue scheduling see section 3.

Both approaches have advantages and disadvantages. Those are discussed in section 4. The next section, section 5, describes a design for a scheduling method that combines both approaches into a system satisfying both human and computer driven observatories.

Section 8 describes the implementation of meta-queues scheduling in the RTS2 system. The article concludes with a discussion of results obtained with meta-queues scheduling running on the Fred Lawrence Whipple Observatory (FLWO) 1.2m telescope.[6]

2. MERIT FUNCTION SCHEDULING

Merit function scheduling uses a function to calculate each target's immediate benefit for an observing program. The merit of an observation can be difficult to estimate, and this is one of the major problems of merit-function scheduling.

Let's assume observers would like to observe a target which is at the moment closest to the zenith. Then the merit function, $meritf$, is:

$$meritf(T, JD, observer) = altitude(T, JD, observer)$$

where:

T is a target

JD is the current Julian date

$observer$ is the observer position (longitude and latitude)

$altitude(T, JD, observer)$ is a function returning a target's T altitude as seen from the site on Earth at the $observer$'s coordinates at date JD

Merit functions are adapted to the needs of the individual observatories. Various terms and parameters can be added so the resulting schedule would match observer's wishes. The current RTS2 merit function is:

$$\begin{gathered} meritf(T, JD, observer, ha, Ld, lo, ldo) = priority(T) + bonus(T) \\ +altitude(T, JD, observer) * 2 \\ +log((180 - ha)/15.0) \Leftrightarrow (ha < 165) + log((ha - 180)/15.0) \Leftrightarrow (ha > 195) \\ -log(61 - Ld) \Leftrightarrow (Ld < 60) - log(lo/3600) * 50 \Leftrightarrow (lo > 3600 \wedge lo < (3 * 3600) \\ -log(3) * 50 \Leftrightarrow (lo < 86400) - sin(ldo * \pi/4) * 5 \end{gathered}$$

Where, in addition to parameters described above and common mathematical symbols (log, sin and π):

$priority(T)$ is the current target T priority

$bonus(T)$ is the current target T bonus

ha is the hour angle for target T as observed from the site on Earth at the $observer$'s coordinates at date JD

Ld is the target lunar distance in degrees

lo is the difference in seconds between the current time (JD) and the last observation time for the target

ldo is the number of target T observations in 24 hours preceding date JD

$\Leftrightarrow$ is the logical iff expression. The value on the right will be used only if conditions on the left are satisfied. For example, $log((180 - ha)/15.0) \Leftrightarrow (ha < 165)$ evaluates to 0 if $ha >= 165$

There are other possible implementations of merit functions. The expression can include the current site seeing, moon phase, number of good images of the target or time left until the end of period during which the target can be observed. Expressions can include target-dependent multiplication factors, or weights. Thus, some targets may be configured with a high relative weight on one expression, while others can decide to ignore this expression.

Given a merit function, the algorithm for target selection is simple. It finds the target with maximum value of the merit function for given instant of all targets, and selects this target as the one to be observed.

Various methods can be used to optimise merit function scheduling. For example, the RTS2 merit function $meritf$ uses an expression parametrised with lo to force the selection of unobserved targets for observation. Besides this, the target properties can be set so the target will be removed from the set of targets evaluated for autonomous selection as soon as it is observed - either indefinitely[*] or for a given time[†].

The more complex the merit function becomes, the more complex it is to evaluate which targets will be selected and how the function will behave in the future. One can add multiple functions to the expression, but all those will just turn the selector behaviour even more obscure. The next paragraph discusses how the scheduling can be simplified by allowing multiple criteria to be considered.

2.1 Multiple objective global optimisation

As it was illustrated in the previous example, merit functions can become quite complex. This is partly due to the need to weight different factors leading to a single number representing a target's merit. As scheduling is in the NP-hard[7] class of problems, finding the best solution requires traversal of the full solution space – with a size that is exponential to the size of input target set. So the usual merit function implementations are short-sighted - only a handful of the successive targets can be considered. It is then hard to create a night plan using just a merit function. If a night plan must be created, some heuristic is usually adopted – for example targets are ordered in west–east direction, as targets near the west horizon should be observed as early as possible before they set, and targets to the east later in the night as Earth rotation places them closer to zenith.

Various strategies can be employed to simplify complex merit functions. Some expressions can be written as constraints, requiring targets to match certain conditions to be included in the target set considered for selection. One of a very good example of such constraint can be zenith distance limit – targets below pointing limits of the telescope shall not be considered for merit function evaluation.

The Master's thesis "Genetic Algorithm for Robotic Telescope Scheduling"[8] discusses design and implementation of a multiple objectives scheduling algorithm which uses Non-dominated Sorting Genetic Algorithm II (NSGA II).[9] This algorithm works by optimising a global observing schedule with NSGA II. Instead of relying on a single merit function, the algorithm works with multiple functions and tries to optimise all of them. This approach further reduces the complexity of the scheduling.

The genetic algorithm is inspired by natural selection process. The GA scheduling works by representing possible schedules as *chromosomes*. When algorithm starts, various possible schedules are generated for population zero. During virtual evolution, the *chromosomes* are *crossed*, *mutated* and *repaired*. The best are *selected* to form next generation. Instead of exploring full solution space, genetic algorithm explores only a small fraction of it. This allows it to effectively search solution for NP-hard, exponential size problems.

[*]targetdisable script command

[†]tempdisable script command

NSGA II scheduling can provide multiple possible schedules for the night. An experienced observer can then choose the one which best matches the expectations. This approach was recently chosen by CFHT for implementing the intelligent scheduler running on top of their queue system.[10]

While NSGA II optimisation looks promising, its inputs are difficult to explain to the observers. Experienced observers prefer to simply provide a list of targets they would like to observe, combine it with targets proposed by other observers and let that schedule run. Queue scheduling, discussed in the next chapter, is much better for direct interaction of investigators with the scheduling.

3. QUEUE SCHEDULING

"In computer science, a queue is a particular kind of abstract data type or collection in which the entities in the collection are kept in order and the principal (or only) operations on the collection are the addition of entities to the rear terminal position and removal of entities from the front terminal position. This makes the queue a First-In-First-Out (FIFO) data structure. In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that once an element is added, all elements that were added before have to be removed before the new element can be invoked. A queue is an example of a linear data structure."[11]

Scheduling queue entities are targets which should be observed. When the next target should be observed, it is removed from the top of the queue. Targets are usually added to the end of the queue, but it is reasonable to assume that observers may like to change the order of the queue targets. It is better to think about a scheduling queue in a different manner than in the definition given above. The scheduling queue can be for our purposes defined as an ordered set, with the following operations: remove top entity, insert entity to arbitrary position, change set order, delete an entity, repeat as needed.

Queue scheduling can be depicted on a computer screen in a diagram, showing rectangles representing targets on the time axis (as can be seen in figure 1). Queue scheduling is mainly used in service mode observing at large observatories. Service mode means that the observations are done by observatory staff on behalf of investigators requesting the observation, who are not present during the observation. Trained observers spend each night at the telescope and work off a list of targets to observe together with required instrument setup. The targets are put into a queue, from which the observatory control system picks a target, sets up instruments for its observations, and commands exposures as required by the scheduling entry. The observer then just monitors the system and troubleshoots any problems.

4. ADVANTAGES AND DISADVANTAGES OF CLASSICAL SCHEDULING

Following table compares queue and merit function scheduling. As it was discussed in previous chapters, the main difference is that merit function scheduling is good for autonomous observatories, while queues are good for trained observers.

Table 1. Comparison of queue and merit function scheduling

Feature	Merit function	Queues
Investigator friendly	bad, as concept of merit functions is hard to explain and understand	moderate, queue combined with graphical user interfaces are relatively simple to explain
Flexibility	good, as merit function can be quickly recalculated with different input (seeing,..) parameters	bad, as it requires significant user interaction with the system
Automatic interruptions, ToOs	yes, new merit function can be calculated for next observation after ToO	yes, but can destroy end of the queue (targets becoming unobservable due to setting below pointing limits,..)
Autonomous operation	excellent, as merit function is a simple algorithm which can be robustly programmed	bad, queue requires human interaction – investigators must fill the queue

5. COMBINING QUEUE AND MERIT FUNCTION SCHEDULING

Merit function scheduling excels where queue scheduling fails and vice versa. Given this, the idea to integrate both scheduling approaches into a common framework seems very promising.

What if the queue system is to be used for obtaining user-requested observations, while merit function scheduling is to be used when there are no entries in the queue system to obtain service-type observations of targets? This is a pseudocode representation of such an approach:

```
while (canObserve ())
{
   if (queueNotEmpty ())
      t = selectFromQueue ();
   else
      t = selectFromMeritFunction ();
   observe (t);
}
```

While this approach will work, it will not address the inflexibility of queue scheduling. There is a selection coming from a single queue, which in its raw base algorithm allows just for the top target to be removed by the **selectFromQueue** procedure. So either the system will pick a target from the queue if the queue is not empty, or it will select a target using merit function evaluation.

To resolve this, one can slightly modify a queue selection algorithm in the **selectFromQueue** method. The modified algorithm should select a target only if the target should be observed. There can be multiple criteria for when the target should be observed – starting with a flag indicating whether the queue scheduling is enabled at all, including the visibility of the target and checking whether the target meets additional constraints, such as a particular time during which it should be observed.

Given that there is an algorithm for a single queue with the properties given above, it is trivial to extend the system to multiple queues. The scheduling algorithm loops over all queues, requesting the next target from each queue. If a target is found, the loop terminates and the target is observed. If queue scheduling cannot find a target to observe, then merit function scheduling is asked to provide one. A pseudo-code representation of such an algorithm is:

```
while (canObserve ()) {
   for (q in queues) {
      t = selectFromQueue (q);
      if (t)
         break;
   }
   if not (t) {
      t = selectFromMeritFunction ();
      if not(t) {
         debug ("cannot find target for observation. 30 sec sleep");
         sleep (30);
         continue;
      }
   }
   observe (t);
}
```

This approach allows investigators to put their targets into their queue. Either a night observer or an autonomous system can then disable or enable queues, based on predefined constraints. Using such an approach, queues become flexible, allowing for simple change of the path the scheduling will take during the night.

We will call the structure holding multiple queues with special characteristics **meta-queues**. Use of meta-queues for scheduling is discussed in the next section.

6. META-QUEUES SCHEDULING

As discussed in section 3, the designation ”queue scheduling” is somewhat misleading and might be renamed to ”scheduling from an ordered set of targets.”

RTS2 queues become even more complex than the ordered set. First, as discussed in the previous section, there are multiple queues – hence the plural in the name.

Second, a queue has an associated queue type. The queue type can change how each queue is ordered and what happens during and after a target is selected from the queue. For details please see discussion in section 6.1.

Third, the queue member is a structure, containing a pointer to the target and optional start and end times of required target observations. With this extra information, a member of the queue can be removed if its end time expires. A queue may also be put on hold when the start time of target on the top position is in the future.

The algorithm for selection from the queues can be written in a pseudo-code as:

```
proc selectFromQueue (q, &maxDuration) {
   filterExpired (q);
   filterUnobservable (q);
   proposed = selectNext (q, now, maxDuration);
   if (proposed) {
      markSelected (q);
      return proposed;
   }
   if (startTime (q) - now < maxDuration)
      maxDuration = startTime (q) - now
}
```

filterExpired removes from the queue all targets with end time in the past. This guarantees that the requests with target time expired will not be scheduled. It also removes targets which are in the queue before a target with start time in the past. This guarantees that if there is a target in the queue which should be observed now, it will be observed at the expense of targets ahead of it. This is important to allow for time-critical observations to start at a required user-specified time, even if the queue execution was delayed, for example by inclement weather.

filterUnobservable removes from the top of the queue targets which cannot be observed at the moment. Based on the queue parameters, those targets are either moved after the first target in the queue which can be observed at a given moment, or completely removed from the queue. This guarantees that the queue will not become blocked by the targets entered into the queue which due to any delay set before they can be observed.

selectNext returns the queue top target if and only if the queue top target start time is either not specified or in the past.

startTime returns the expected start time of the top target from the queue, or not-a-number if there is no start time attached to the top target. **now** is the current time.

maxDuration is used to constrain the maximal duration of execution of the selected target. If the expected execution duration of a target extends past the available time, the target is not considered for selection. **maxDuration** is set by a procedure that calculates the time available until dawn, and updated if it is lower than the start time of the first observation from the queue. The queue system guarantees that if there is a queue with the target which observations should be started later, targets from queues considered after this queue will finish before the given queue start time is reached.

6.1 Meta-queues Ordering and Selection Algorithms

Meta-queues can be configured to order and select targets in a different fashion to the standard first-in-first-out (FIFO) algorithm. The available variations are:

CIRCULAR similar to **FIFO**, but places selected targets at the end of the target queue. In this way, queues can be configured to continuously observe a set of targets throughout the night.

HIGHEST order targets by altitude.

WEST–EAST order targets from west to east. This allows for targets on the west to be observed first.

WEST–EAST–MERIDIAN order targets west of the meridian by priority and east from meridian the same way as the **WEST–EAST** queue. This is to allow for high priority targets west of the meridian, meaning those targets that are currently setting, so the highest priority targets will be observed first.

OUT–OF–LIMITS order targets by remaining time until they set below observing constraints.

A user or program can change the queue type anytime. For example, one can change a queue from **FIFO** to **WEST–EAST–MERIDIAN** if bad weather is approaching. This allows for targets with the highest priority, which are setting, to be observed first.

Circular queue can be used to monitor multiple targets throughout a night. Assume we have targets A,B,C, which should vary on time scales long enough to allow for some gaps between target visits. Assuming A, B and C are visible, the scheduler will repeat sequence A,B,C. Targets that are not visible will be skipped.

6.2 Target of opportunity interruptions

While the previous section deals with making the queues flexible, it does not describe how this flexibility can be used for target of opportunity observations.

Targets of opportunity can be introduced into the RTS2 target database with custom clients, which are available for GCN[12] and Pierre-Auger high energy showers. The current procedure is to call the ”now” command of the executor to immediately execute visible targets of opportunity, and raise the target priority so that the merit function will pick the target for the next observations. While the ”now” part worked without problems and there is no need to change it, the follow-up planning with the merit function was difficult.

The preferred approach with the queue scheduling for the follow-up observations is to put targets of opportunity into a special queue. This queue can be disabled if the observers do not want to interrupt their observing plan. The target script can add targets to any queue if more observations are desired.

7. PLANNING OF THE NIGHT WITH META-QUEUES SCHEDULING

Observers can interact with the scheduling subsystem either through a graphical user interface, or from a command line. Targets must be entered into the database before they are scheduled with various target management tools that RTS2 provides. Users can append targets into a queue, move targets to an arbitrary position, remove targets from the queue or clear the queue.

The scheduling subsystem includes a scheduling simulator. The simulator uses queue entries to produce a simulated night run, showing which targets will be executed during the night[‡]. The simulator can be used by users to confirm that the entries they insert in the queues will be executed as desired.

Different queues can be populated by different users. For example, occultation targets requiring execution at a given time can be put into one queue, and supernova follow-up targets into another. Observers, if awake during the night, can monitor the night run and disable the queue if conditions do not allow target observations.

[‡]assuming ideal run conditions – i.e., a night without any interruptions caused by bad weather or instrument failure

Queue	Target	Start date and time	End date and time	Script duration
transit	TR 1	29th December 2012, 21:30	29th December 2012, 23:00	1m
(FIFO)	TR 2	30th December 2012, 23:30	31st December 2012, 09:00	2m
	TR 3		31st December 2012, 23:00	1m
service	SN A			50m
(FIFO)	SN B			
	....			
	SN M			
	SN N	1st January 2013, 00:20		
	Blazar A	1st January 2013, 02:30		
	Blazar B			
	Blazar C			
backup	B 1			
(circular)	B 2			20m
	B 3			

Table 2. Example schedule combining transits, service and backup observations

Transit queue targets will not be remove before they *end time*, as the queue is marked as *Keep observed*. Service queue target will be considered only if there aren't targets in transit queue, backup queue targets only if there aren't targets in transit and service queues.

7.1 Example schedule

Let's assume an investigator would like to observe three transits, named *TR 1*, *TR 2* and *TR 3*. The transits must be observed on nights from 29th December 2012 to 1st January 2013.

And let's assume another investigator would like to observe supernovae named *SN A*, *SN B* through *SN N*, and *Blazar A* through *Blazar C*. *SN A* through *SN M* can be observed anytime, but *SN N* observations must start nearest 1st January 2013 at 00:20. If any of *SN A* through *SN M* are not observed before *SN N*, they should be removed from the schedule. Similarly, *Blazar A* observations must start nearest 1st January 2013 at 02:30.

On top of that, there is "backup" queue containing targets to be observed while there is nothing else to do. This queue is of **CIRCULAR** type.

Table 2 shows the queue setup for such observations. Table 3 shows how those queues can turn into observations with a period of inclement weather. Figure 1 depict how the queues user interface would present an user with the queues setup.

8. 1.2M FLWO TELESCOPE IMPLEMENTATION

Of the telescopes which uses RTS2, the 1.2m FLWO telescope is a primary user of queue scheduling. The telescope is a special case: due to the complexity and availability of the existing observatory control system, with which both local and remote observers are well familiarised, RTS2 is used only for autonomous control – scheduling, preparation and execution of the observation scripts, weather monitoring and accounting of the autonomous observations. RTS2 does not have direct control of the hardware. It communicates with the hardware through the FLWO Telshell, a modified tcsh[§]. This architecture is depicted on figure 2.

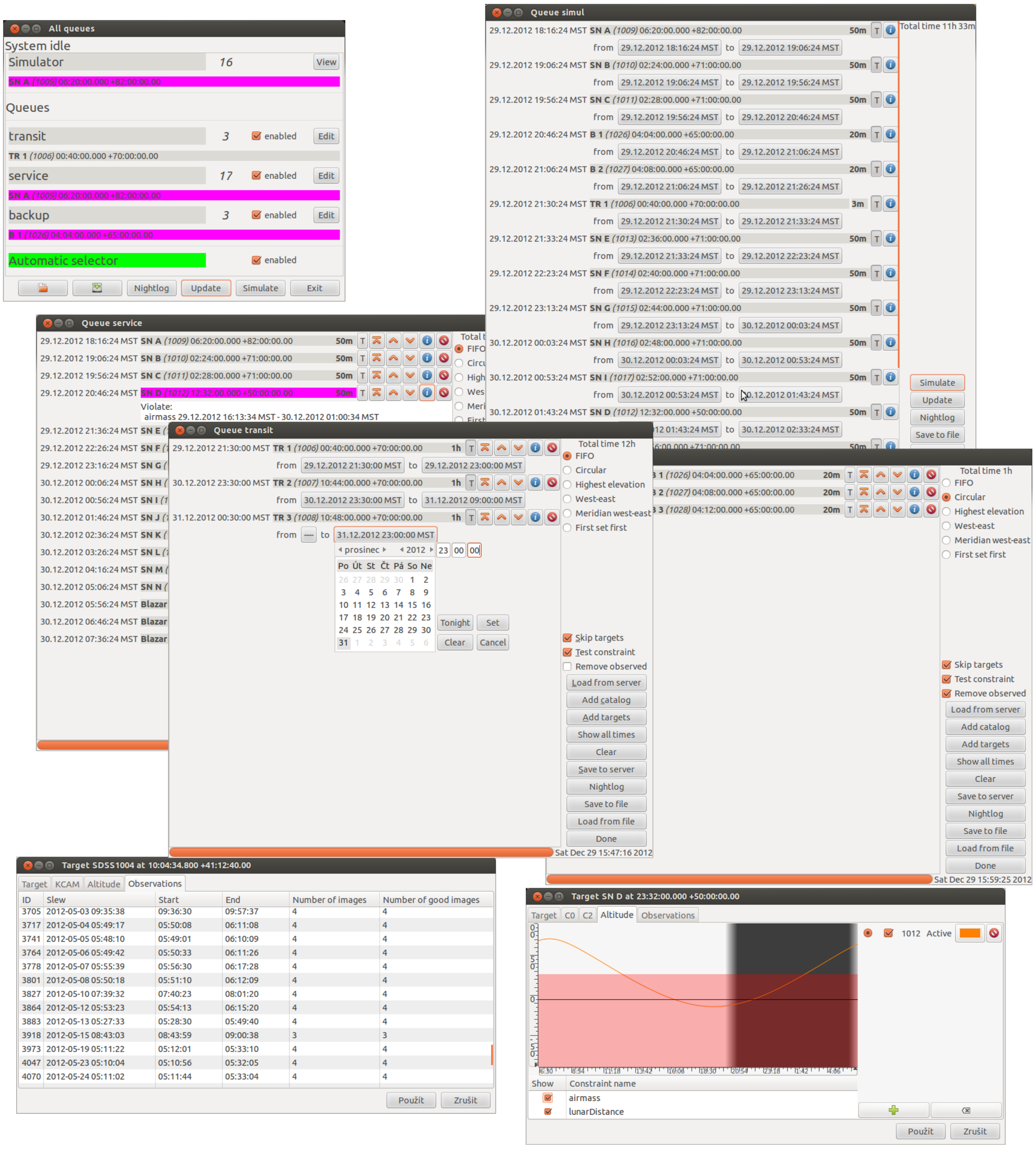
The scheduler running as **rts2-selector** of the RTS2 continuously evaluates queues and merit functions and produces as a single output the next target which should be observed. The target number is retrieved through a JSON[13] interface by a script running inside **Telshell**. The script retrieves instructions for instrument setups as an XML file from the database, and translates this XML file into the script for the observation. Then the observation script is sourced.

The RTS2 processes run as daemons in the background. The scripts running in the Telshell query the weather state and readiness of the system, and take appropriate action if bad weather is detected. If an investigator would

[§]the modification allows control of the hardware through special shell commands

Description	Start date and time	End date and time
day		29th December 2012, 18:16
SN A	29th December 2012, 18:16	19:06
SN B	19:06	19:56
SN C	19:56	20:46
B 1 *(as SN D is too long)*	20:46	21:06
B 2	21:06	21:26
merit scheduling or idle	21:26	21:30
TR 1	21:30	23:00
SN E *(SN D is not visible)*	23:00	23:50
SN F *(SN D is not visible)*	23:50	30th December 2012, 00:20
bad weather – idle	30th December 2012, 00:20	04:23
SN D	04:23	05:13
SN G	05:13	06:03
B 3 *(to fill time)*	06:03	06:23
merit scheduling or idle	06:23	06:34
(day)	06:34	18:17
bad weather	18:17	20:34
SN H	20:34	21:24
SN I	21:24	21:26
remote session	21:26	23:06
B 1	23:06	23:26
merit scheduling or idle	23:26	23:30
TR 2	23:30	31th December 2012, 01:30
observer disables transit and service queues due to inclement weather		01:28
B 2	31th December 2012, 01:30	01:50
B 3	01:50	02:10
B 1	02:10	02:15
bad weather	02:15	04:35
observer enables service queue		04:25
SN J	04:35	05:25
observer enables transit queue		05:15
TR 2	05:25	06:34
(day)	06:34	18:17
TR 3	18:17	23:00
SN K	23:00	23:50
B 2	23:50	1st January 2013, 00:10
merit scheduling or idle	1st January 2013, 00:10	00:20
SN N *SN L and SN M are removed from the queue*	00:20	01:10
B 3	01:10	01:30
B 1	01:30	01:50
B 2	01:50	02:10
B 3	02:10	02:30
Blazar A	02:30	03:20
Blazar B	03:20	04:10
Blazar C	04:10	05:00
B 1	05:00	05:20
B 2	05:20	05:40
B 3	05:40	06:00
B 1	06:00	06:20
merit schedule or idle	06:20	06:34
(day)	06:34	

Table 3. Simulation of observations resulted from the example schedule described in table 2.
This table shows important events happening between 29th December 2012 and 1st January 2013. This is assuming the SN and Blazar target observations take their allotted time. In a real run, the time will differ as targets will use different scripts and due to the different distances the telescope has to slew between targets.



```
All queues
System idle
Simulator                16      View
SN A (1009) 06:20:00.000 +82:00:00.00
Queues
transit                  3   enabled   Edit
TR 1 (1006) 00:40:00.000 +70:00:00.00
service                  17  enabled   Edit
SN A (1009) 06:20:00.000 +82:00:00.00
backup                   3   enabled   Edit
B 1 (1026) 04:04:00.000 +65:00:00.00
Automatic selector           enabled
Nightlog  Update  Simulate  Exit
```

```
Queue simul
29.12.2012 18:16:24 MST SN A (1009) 06:20:00.000 +82:00:00.00     50m   Total time 11h 33m
    from 29.12.2012 18:16:24 MST to 29.12.2012 19:06:24 MST
29.12.2012 19:06:24 MST SN B (1010) 02:24:00.000 +71:00:00.00     50m
    from 29.12.2012 19:06:24 MST to 29.12.2012 19:56:24 MST
29.12.2012 19:56:24 MST SN C (1011) 02:28:00.000 +71:00:00.00     50m
    from 29.12.2012 19:56:24 MST to 29.12.2012 20:46:24 MST
29.12.2012 20:46:24 MST B 1 (1026) 04:04:00.000 +65:00:00.00      20m
    from 29.12.2012 20:46:24 MST to 29.12.2012 21:06:24 MST
29.12.2012 21:06:24 MST B 2 (1027) 04:08:00.000 +65:00:00.00      20m
    from 29.12.2012 21:06:24 MST to 29.12.2012 21:26:24 MST
29.12.2012 21:30:24 MST TR 1 (1006) 00:40:00.000 +70:00:00.00     3m
    from 29.12.2012 21:30:24 MST to 29.12.2012 21:33:24 MST
29.12.2012 21:33:24 MST SN E (1013) 02:36:00.000 +71:00:00.00     50m
    from 29.12.2012 21:33:24 MST to 29.12.2012 22:23:24 MST
29.12.2012 22:23:24 MST SN F (1014) 02:40:00.000 +71:00:00.00     50m
    from 29.12.2012 22:23:24 MST to 29.12.2012 23:13:24 MST
29.12.2012 23:13:24 MST SN G (1015) 02:44:00.000 +71:00:00.00     50m
    from 29.12.2012 23:13:24 MST to 30.12.2012 00:03:24 MST
30.12.2012 00:03:24 MST SN H (1016) 02:48:00.000 +71:00:00.00     50m
    from 30.12.2012 00:03:24 MST to 30.12.2012 00:53:24 MST
30.12.2012 00:53:24 MST SN I (1017) 02:52:00.000 +71:00:00.00     50m   Simulate
    from 30.12.2012 00:53:24 MST to 30.12.2012 01:43:24 MST               Update
30.12.2012 01:43:24 MST SN D (1012) 12:32:00.000 +50:00:00.00     50m   Nightlog
    from 30.12.2012 01:43:24 MST to 30.12.2012 02:33:24 MST               Save to file
```

```
Queue service
29.12.2012 18:16:24 MST SN A (1009) 06:20:00.000 +82:00:00.00   50m
29.12.2012 19:06:24 MST SN B (1010) 02:24:00.000 +71:00:00.00   50m
29.12.2012 19:56:24 MST SN C (1011) 02:28:00.000 +71:00:00.00   50m
29.12.2012 20:46:24 MST SN D (1012) 12:32:00.000 +50:00:00.00   50m
    Violate:
    airmass 29.12.2012 16:13:34 MST - 30.12.2012 01:00:34 MST
29.12.2012 21:36:24 MST SN E
29.12.2012 22:26:24 MST SN F
29.12.2012 23:16:24 MST SN G
30.12.2012 00:06:24 MST SN H
30.12.2012 00:56:24 MST SN I
30.12.2012 01:46:24 MST SN J
30.12.2012 02:36:24 MST SN K
30.12.2012 03:26:24 MST SN L
30.12.2012 04:16:24 MST SN M
30.12.2012 05:06:24 MST SN N
30.12.2012 05:56:24 MST Blazar
30.12.2012 06:46:24 MST Blazar
30.12.2012 07:36:24 MST Blazar
```

```
Queue transit
29.12.2012 21:30:00 MST TR 1 (1006) 00:40:00.000 +70:00:00.00   1h   Total time 12h
    from 29.12.2012 21:30:00 MST to 29.12.2012 23:00:00 MST          (●) FIFO
29.12.2012 23:30:00 MST TR 2 (1007) 10:44:00.000 +70:00:00.00   1h   ( ) Circular
    from 29.12.2012 23:30:00 MST to 31.12.2012 09:00:00 MST          ( ) Highest elevation
31.12.2012 00:30:00 MST TR 3 (1008) 10:48:00.000 +70:00:00.00   1h   ( ) West-east
    from — to 31.12.2012 23:00:00 MST                                ( ) Meridian west-east
                                                                     ( ) First set first
    prosinec  2012   23 00 00
    Po Út St Čt Pá So Ne
    26 27 28 29 30  1  2
     3  4  5  6  7  8  9
    10 11 12 13 14 15 16
    17 18 19 20 21 22 23   Tonight  Set
    24 25 26 27 28 29 30
    31  1  2  3  4  5  6   Clear  Cancel
[✓] Skip targets
[✓] Test constraint
[ ] Remove observed
Load from server | Add catalog | Add targets | Show all times | Clear | Save to server | Nightlog | Save to file | Load from file | Done
Sat Dec 29 15:47:16 2012
```

```
Queue backup
B 1 (1026) 04:04:00.000 +65:00:00.00   20m   Total time 1h
B 2 (1027) 04:08:00.000 +65:00:00.00   20m   ( ) FIFO
B 3 (1028) 04:12:00.000 +65:00:00.00   20m   (●) Circular
                                             ( ) Highest elevation
                                             ( ) West-east
                                             ( ) Meridian west-east
                                             ( ) First set first
[✓] Skip targets
[✓] Test constraint
[✓] Remove observed
Load from server | Add catalog | Add targets | Show all times | Clear | Save to server | Nightlog | Save to file | Load from file | Done
Sat Dec 29 15:59:25 2012
```

Target SDSS1004 at 10:04:34.800 +41:12:40.00

Target | KCAM | Altitude | Observations

ID	Slew	Start	End	Number of images	Number of good images
3705	2012-05-03 09:35:38	09:36:30	09:57:37	4	4
3717	2012-05-04 05:49:17	05:50:08	06:11:08	4	4
3741	2012-05-05 05:48:10	05:49:01	06:10:09	4	4
3764	2012-05-06 05:49:42	05:50:33	06:11:26	4	4
3778	2012-05-07 05:55:39	05:56:30	06:17:28	4	4
3801	2012-05-08 05:50:18	05:51:10	06:12:09	4	4
3827	2012-05-10 07:39:32	07:40:23	08:01:20	4	4
3864	2012-05-12 05:53:23	05:54:13	06:15:20	4	4
3883	2012-05-13 05:27:33	05:28:30	05:49:40	4	4
3918	2012-05-15 08:43:03	08:43:59	09:00:38	3	3
3973	2012-05-19 05:11:22	05:12:01	05:33:10	4	4
4047	2012-05-23 05:10:04	05:10:56	05:32:05	4	4
4070	2012-05-24 05:11:02	05:11:44	05:33:04	4	4

Použít Zrušit

Target SN D at 23:32:00.000 +50:00:00.00

Target | C0 | C2 | Altitude | Observations

1012 Active

Show	Constraint name
✓	airmass
✓	lunarDistance

Použít Zrušit

Figure 1. Queue GUI

User scheduling interface showing queues setup from the example described in section 7.1. On image top are windows showing queues available at the system and current selection from the queues. Right is simulation showing how the system thinks the observations will be executed. At middle of the image are shown service queue filled with supernova observations, transit queue demonstrating how user can enter queue times, and backup queue with **CIRCULAR** queue ordering. Windows showing observing logs of a target and altitude plot of another target are visible on image bottom.

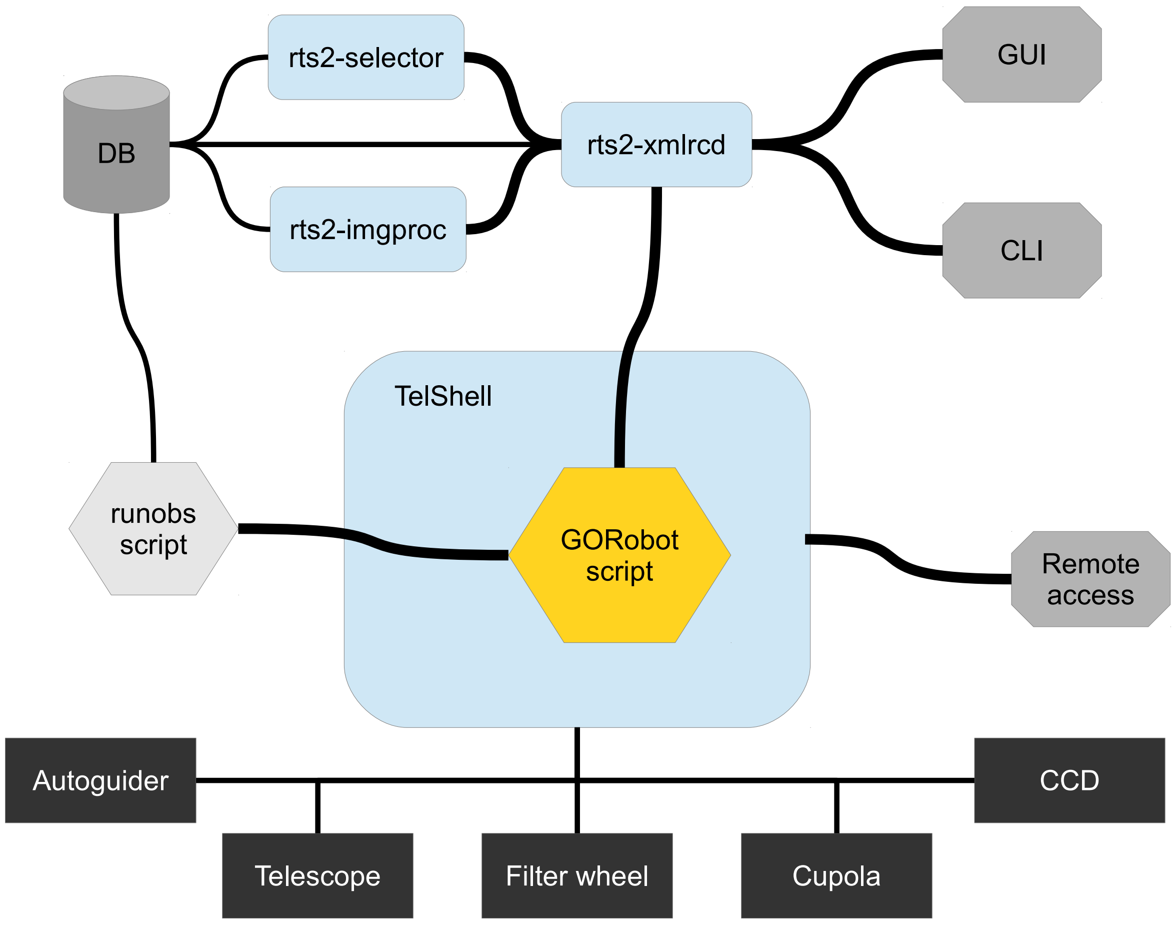


Figure 2. Integration of FLWO 1.2m telescope TelShell with RTS2
Telshell controls observatory hardware through various interfaces. It provides commands to access the hardware. Database stores targets data and robot observations logs. *rts2-selector*, *rts2-imgproc* and *rts2-xmlrpcd* are RTS2 processes responsible for scheduling, image processing and JSON access to the RTS2 environment. Graphical user interface (GUI) and command line interface (CLI) interacts with *rts2-xmlrpcd* to access and change robot parameters. Remote access is possible by stopping **GORobot** script and typing commands directly in *TelShell*.

like to take over and observe using commands typed into Telshell, she can interrupt the robot script and take over. Once the investigator is done with manual control, she can switch observations back to RTS2 by running **GOrobot** command.

An example of a complete flow of commands needed for construction of the script and their outputs (which are used as inputs for the next command) is summarised in table 4. It should be noted that the target script command is automatically repeated if the current target is proposed by scheduler as the next target. Commands such as turning on/off the autoguider are ignored if the state is equal to the commanded state. The system loses about 0.2 seconds between consecutive exposures of the same target coming from different script execution.

Command	Output, comments
rts2-json -G SEL.next_id	1000 - identification of the next target
rts2-targetinfo –parse KCAM 1000	`<script device='KCAM'>` `  <set device='KCAM' value='ampcen' op='=' operands='2'/>` `  <set device='KCAM' value='autoguide' op='=' operands='ON'/>` `  <tempdisable>1800</tempdisable>` `  <set device='KCAM' value='filter' op='=' operands='i'/>` `  <exposure length='80'/>` `</script>`
xsltproc flwo.xsl -	`...  script head skipped, only an example is given` `if ( $? == 0 && $in == 1 ) then` `   rm -f $lasttarget` `   set continue=0` `   rts2-logcom "Interrupting target $name ($tar_id)"` `endif` `ccd gowait $exposure` `...`

Table 4. Commands used for the 1.2m FLWO telescope scheduling system

9. RESULTS

The FLWO robot project was started during fall 2010. Table 5 shows 1.2m FLWO telescope statistics from late 2010 when the robot project was first tested on the telescope up to the end of May 2012.

In the statistics table, the column marked "Nights" shows the number of nights the robot was active. "Sky time" is the sum of exposure times of all images acquired by the robot at the given month. Observ. is the number of robot observations, images the number of images acquired by the robot.

Nights and total time are numbers of nights and observing time extracted from the 1.2m logs. As those are filled by observers and don't account for dead time lost by telescopes slews and detector readout, it is expected the "Total nights" will never equal "Sky time". Depending on the lengths of exposures performed during the night, "total time" can include from 10% to 50% or even more dead time. As an extreme, dead time can be 140% for 10 second exposures – as the detector readout time is 14 seconds.

Nights is the percentage of nights the robot was used at least for part of each night. Time is the fraction of sum of sky time, number of observations multiplied by 90 and number of images multiplied by 14 divided by the total time. The expected median slew time is the 90 seconds and 14 seconds is the budget for detector readout.

Although time share cannot be precisely determined, as the current system does not allow to differentiate between images taken by the robot and human observers and there are no statistics of the images and observations performed by the local or remote observers, there is a clear trend of increased use of the robot for observations from late 2011. This is correlated with improvements of the autoguider, scheduling user interfaces and familiarity of the investigators with the queue scheduling.

Month	Robot				Observer's log		Robot share	
	Nights	Sky time	Observ.	Images	Nights	Total time	Nights	Time
10/2010	4	4:23:10	17	58	28	195:30:00	14.3%	2.6%
11/2010	3	3:02:00	12	46	30	248:45:00	10.0%	1.4%
12/2010	4	19:56:30	44	247	28	219:30:00	14.3%	10.0%
01/2011	6	24:19:40	84	293	30	311:30:00	20.0%	8.8%
02/2011	4	12:29:31	47	179	26	224:00:00	15.4%	6.4%
03/2011	5	27:35:00	78	383	29	289:05:00	17.2%	10.7%
04/2011	4	11:32:38	80	215	29	247:45:00	13.8%	5.8%
05/2011	5	26:21:37	99	419	29	220:00:00	17.2%	13.8%
06/2011	1	2:42:00	11	36	29	214:00:00	3.4%	1.5%
07/2011	-	-	-	-	22	98:15:00	-	-
08/2011	-	-	-	-	1	8:00:00	-	-
09/2011	8	20:58:31	95	589	23	127:54:00	34.8%	20.0%
10/2011	16	80:16:42	306	2187	31	262:00:00	51.6%	36.8%
11/2011	11	57:53:07	215	2254	26	202:48:00	42.3%	35.5%
12/2011	19	99:42:55	430	4014	20	169:00:00	95.0%	74.6%
01/2012	25	114:05:29	416	5533	25	257:40:00	100.0%	56.7%
02/2012	22	97:46:28	449	3100	23	214:00:00	95.7%	56.6%
03/2012	18	94:58:26	362	4674	26	234:00:00	69.2%	52.2%
04/2012	29	153:58:31	579	7679	31	244:00:00	93.5%	81.3%
05/2012	31	157:40:38	571	6134	31	236:55:00	100.0%	82.6%

Table 5. FLWO 1.2m Robot statistics November 2010 - May 2012
Statistics of the autonomous observations of the 1.2m FLWO telescope. Nights, Sky time and number of observations and images are extracted from the robot database. Nights and Total time are counted from observing logs. Robot shares are computed as the percentage of nights and time the robot was active from the nights recorded in the observers' logs. To adjust for CCD readout and telescope slew, an estimated dead time is added to the sky time before the fraction of total time is calculated.

10. CONCLUSIONS

The article presented two basic approaches for the scheduling of observations – queue and merit function scheduling. It shows how the two can be combined into "meta-queue" scheduling, which allows observers to easy schedule their targets. Use of the described design for scheduling of the actual observatories demonstrates the system performs well. It also demonstrates how manually controlled observations can co-exist with autonomous either pre-programmed or dynamic observing schedules.

Further work is needed, particularly on the user interface and presentation of the collected statistics. As the system stores all data in the database, we are confident the extensions will be possible and allow us to better understand how the system behaves and what should be improved for it to work even better.

ACKNOWLEDGMENTS

We would like to acknowledge support from the FLWO staff and patience of the first users who had to find the initial bugs in the system. This research was funded by SAO grants. PK and MP would like to acknowledge grant ME09052 of program KONTAKT of the Ministry of Education, Youth and Sports of the Czech Republic and EU grant project GLORIA No. 283783 (FP7-Capacities). The design of the scheduling mode we presented here was inspired and influenced by the authors' familiarity with scheduling at other observatories. As it would

be difficult to provide a full list of those observatories, we would like to particularly mention inspiration from Gemini for multiple queues system and STELLA for merit function scheduling.

REFERENCES

[1] Johnston, M. D. and Miller, G. E., "Spike: Intelligent scheduling of hubble space telescope observations," in [*Intelligent Scheduling*], 391–422, Morgan Kaufmann Publishers (1994).

[2] Granzer, T., Weber, M., and Strassmeier, K. G., "Three Years of Experience with the STELLA Robotic Observatory," *Advances in Astronomy* **2010** (2010).

[3] Fraser, S. N., "Scheduling for Robonet-1 homogenous telescope network," *Astronomische Nachrichten* **327**, 779 (Sept. 2006).

[4] Puxley, P. and Jørgensen, I., "Five years of queue observing at the Gemini telescopes," in [*Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*], *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* **6270** (July 2006).

[5] Chavan, A. M., Giannone, G., Silva, D., Krueger, T., and Miller, G., "Nightly Scheduling of ESO's Very Large Telescope," in [*Astronomical Data Analysis Software and Systems VII*], Albrecht, R., Hook, R. N., and Bushouse, H. A., eds., *Astronomical Society of the Pacific Conference Series* **145**, 255 (1998).

[6] "Flwo 1.2m telescope." http://www.sao.arizona.edu/FLWO/48/48.html.

[7] "Np–hard." http://en.wikipedia.org/wiki/NP-hard.

[8] Kubánek, P., "Genetic algorithm for robotic telescope scheduling," *Master en Soft Computing y Sistemas Inteligentes* , Universidad de Granada, Granada, Spain (2008).

[9] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation* **6**, 182–197 (2002).

[10] Mahoney, W. and Veillet, C., "The use of a genetic algorithm for ground-based telescope observation scheduling," in [*Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*], *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* **8448** (in press).

[11] "Queue." http://en.wikipedia.org/wiki/Queue_(data_structure).

[12] Barthelmy, S. D., "GRB Coordinates Network (GCN): A Status Report," *American Astronomical Society Meeting Abstracts* **202** (May 2003).

[13] "Json - java script object notation." http://www.json.org.