Observation Scheduling for a Network of Small-Aperture Telescopes

A. R. Duncan[A,B]

[A] Mt Kent Observatory, Faculty of Sciences, University of Southern Queensland, Toowoomba QLD 4350, Australia
[B] Automated Patrol Telescopes Australia, Pty Ltd., Brisbane QLD 4001, Australia
Email: roy@apta.net.au

Received 2007 March 20, accepted 2007 May 8

Abstract: Over the past several years, a system for accepting, servicing and returning the results from a large number of imaging requests has been developed for use with automated optical telescopes. One of the primary goals of this project is to increase the accessibility of astronomy to school, college and university students. A key component of this system is a request scheduling engine, which produces schedules for each telescope for its current night. This engine is dynamic, adjusting schedules to accommodate new requests and rescheduling failed requests on a time scale of the order of ten minutes. If a telescope is unavailable for an extended period, imaging requests will be reallocated to other telescopes in the network. Various models of dynamic scheduling are considered, and the current implementation is explored with a number of numerical experiments.

Keywords: methods: numerical — telescopes

1 Introduction

The fields of small-telescope and small-observatory automation have matured greatly in recent years. With the availability of good-quality telescopes, CCD cameras, focusers and associated hardware, the construction and operation of small observatories is becoming ever-more feasible.

With this in mind, a project was commenced to create a system in to which such telescopes could be connected. This system would be able to coordinate observations between the member telescopes of the network.

The design goals for the system were that it should be flexible, scalable and easy to use; it should return images and data rapidly; it should be easy to access, using only simple and readily-available client technology (e.g. a web browser) and standard networking protocols.

It was clear from an early stage that this endeavour would require a flexible and robust scheduling engine. This paper discusses the development of a scheduler capable of coordinating observations amongst a series of small-aperture telescopes, distributed over a wide area network. The broader system, encompassing request injection and lifecycle management, scheduling, image aquisition, data retrieval and dispatch, is known as SAROS (the *Scheduled, Autonomous, Remote Observation System*).

In Section 2, overviews of SAROS and of its control and monitoring systems are provided. This is followed by a discussion of observation scheduling for astronomical telescopes in Section 3. Details of the numerical experiments are described in Section 4, with results and discussion presented in Section 5.

2 The Telescope Network

The scheduling engine discussed in this paper forms a part of a larger suite of software, known as SAROS. This broader system encompasses components to accept and inject observation requests, to store and manage these requests through their lifecycle, and to aquire and dispatch image data in an appropriate and timely manner. It is useful to briefly describe this system, in order to place the scheduling engine in context.

The network currently consists of a number of small-aperture, optical telescopes which can be remotely operated. The system was developed and tested using the observatory 'Rochedale (APTA)', which has IAU designation E25. Instruments from the University of Southern Queensland's Mt Kent Observatory complex are also participating in the network, and soon the network will be opened such that any suitable telescope may participate.

Member telescopes are linked to a central server via appropriate network infrastructure. In particular, the current implementation utilises a TCP/IP network. Communication between the central server and the remote telescopes takes place over RFC-standard protocols, primarily HTTP (Fielding et al. 1999) and HTTPS (Dierks & Allen 1999). The SAROS system has been standardised to perform observations through the use of the ACP telescope control package (Denny 2005), although the system retains the capacity to work with any software which has a workable interface.

The operation of the system is based upon 'logical telescope networks'. In this idea, the full network of telescopes is divided in to a number of separate, logical networks. For example, networks can be created for instruments with wide-angle or narrow-angle fields of view, or for instruments with high-end equipment or large apertures, or for use by particular schools or universities or the general public. Access to these logical networks can be restricted based upon, for example, the characteristics of the user. In this way, an individual, a small astronomy club, a school or a university can each choose to make their telescopes available to their friends, members or students alone, or they could choose to make these instruments available to potentially service requests from anyone.

The system incorporates extensive monitoring and logging of hardware, software and network performance metrics, along with detailed information on weather and the states of systems at the remote sites. SAROS also allows for the scheduled switching and control of remote hardware. Monitoring is performed by a customised version of the open-source package NAGIOS[1], with data logging and visualisation handled by the CRICKET package (Allen 1999). NAGIOS has been configured to send notifications to appropriate people, should adverse events affect a particular system or telescope.

Because observations can be requested by a variety of users, with varying requirements and levels of knowledge and skill, a flexible request submission system has been implemented. Requests can be created by a number of mechanisms, including web-based forms of varying complexity, RTML (Pennypacker et al. 2003), or according to an automated regimen. Currently, only RTML version 2.1 (Pennypacker et al. 2002) is supported, although it is intended that RTML 3 (Hessman 2004) will be supported as this matures. Requests can also be injected directly from external systems, via web services. This latter path may prove useful in integrating with external RTML servers or with other software systems. The request injection methodology is extensible, such that it is straightforward to create new methods by which imaging requests can be injected in to the system (e.g. email, SMS).

Requests may specify the coordinates of an astronomical object, or may simply specify a catalogue or common name. In the latter cases, the target's coordinates are looked up in a 'directory' of astronomical objects. The orbital elements of solar system objects (e.g. comets, asteroids and planets) are automatically updated on a regular basis.

Requests also contain information about their priority. The validity period for a request is typically of the order of a week. The priority of a request, as supplied to the scheduler at any given time, will often increase monotonically over this period. This time-dependence of priority is typically used to ensure that requests with less 'time to live' are treated by the scheduler as higher priority. The absolute priority of a request (i.e. independent of time), is determined by a number of factors. It can be influenced by request preferences, by the (logical) telescope network to which the requst is submitted, by who submitted the request, and by the method used to submit the request.

Once injected, imaging requests and all associated data reside in a central store, where they are made available to the scheduling engine. The schedules subsequently constructed for each telescope by this engine are prosecuted by daemons which assume the role of the observer. The observed images are subsequently retrieved and pass through an analysis pipeline. If the imaging attempt was unsuccessful, the request will be flagged and rescheduled for a subsequent observation attempt. If the request was successful, the image and associated information are appropriately dispatched to the request owner.

The SAROS system contains support for 'request sharing', which can be of significant benefit in a system of this type. This model breaks the one-to-one and onto relationship which typically exists between an image request and the actual telescope observation. For example, a newly-arriving request which is identical to one already in a telescope's queue need not require a separate observation to fulfil. Instead, if appropriate, this new request may be associated with the already-queued observation request. In such a case, a successful observation would then satisfy two, independent requests. In this way, increased numbers of individual requests may be handled by a relatively modest number of telescopes.

For details on how to access the system, see Section 7.

3 Request Scheduling

Scheduling is a generic process which may be applied whenever a scarce resource needs to be utilised by, or shared between, many potential 'users' of that resource. Examples include the use of manufacturing machinery, public transport networks, lecture theatres or the allocation of CPU cycles by an operating system.

As applied to the use of large astronomical facilities, scheduling has long been employed by time allocation committees in order to detail which observing program may make use of a telescope at a particular time. With the exception of 'target of opportunity' observations, such schedules are generally prepared months in advance and, once produced, remain substantially unaltered. In contrast, the scheduling systems developed for smaller-aperture telescopes typically operate with smaller time constants, of the order of 24 hours or less.

A number of groups have developed remote and robotic observing facilities, incorporating request scheduling systems. There are too many papers in this field to cite herein, instead the reader is refered to the contributions and references appearing in Adelman, Dukes, & Adelman (1992), Filippenko (1992), Henry & Eaton (1995), and Oswalt (2003).

Schedulers can be broadly classified in to one of two distict categories, called 'dispatch' and 'optimising'. A dispatch scheduling process is one in which pending observing requests are ranked according to an appropriate metric, and the most favourable request is observed next. This process is then repeated for the next and all

[1] `http://www.nagios.org/`

subsequent requests. Examples of dispatch schedulers are discussed by Edgington et al. (1996), Steele & Carter (1997), and Denny (2004).

In contrast, an optimising scheduling process constructs a schedule for a significant period of time (typically one night or longer). It is more complicated and computationally intensive than an equivalent dispatch process, however, once produced, this schedule can in principle be followed without any further use of the scheduler. An optimising approach will typically work towards maximising some metric of schedule quality, and as such will produce better (in some sense) schedules than a dispatch process. Examples of optimising scheduling algorithms can be found in Richmond, Treffers, & Filippenko (1993), and Drummond et al. (1995).

For the SAROS system, it was judged that an optimising scheduler was more appropriate. There are several reasons for this. First, although requiring more computational effort, an optimising scheduler will in general produce more efficient use of a telescope. Second, the resulting schedule may be made available to users of the system, such that they can be given an idea of when their request is likely to be observed. Third, it is important to support request constraints of the form 'request B must be observed at least n hours after request A'. In the general case, where A and B are to be observed by different telescopes, satisfaction of such constraints is more straightforward with the use of an optimising scheduler.

3.1 Scheduler Characteristics

The system to be implemented placed a number of restrictions on the scheduler design. First, the scheduling system had to react quickly to changing circumstances and unanticipated events. Examples of such events are: the arrival of a new request, the cancellation of an existing request, the reassignment of a request from one telescope to another, failure of an observation or failure of equipment, or a problem with network connectivity. An arriving request should be examined by the scheduler within minutes, and a failed imaging attempt should be rescheduled on a similar timescale. At the same time, because scheduling is a computationally expensive procedure, the scheduler should not regenerate schedules unncessarily.

Second, the scheduler must be able to deal with a large number of requests (potentially many tens of thousands), and generate an appropriate schedule rapidly.

Third, the process of revising a schedule which is currently being prosecuted must not interrupt the observing process. This is especially important in the current implementation, because a schedule may need to be regenerated many times over the course of a night as new requests arrive. If observing were to cease whilst a schedule was rebuilt, this could result in the loss of considerable observing time.

Fourth, the scheduler must be flexible. Different users and requests will have different requirements and preferences. The scheduling engine must be able to satisfy at least a basic set of useful requirements. This will be further considered in the next section.

3.2 Constraints

The construction of an observing schedule for a given telescope may be considered as a constraint satisfaction problem. Fortunately, a large body of literature exists in this field, e.g. Schaerf (1999), Barták (1999, 2002).

The most basic scheduling process for night-time astronomical observations involves the satisfaction of several fundamental constraints. First, the target objects must be above the telescope's local (possibly azimuth-dependent) horizon for the duration of the exposure. Second, the solar zenith angle must exceed some value. An angle of approximately 1.9 rad is typically used to ensure that the sky is sufficiently dark. Third, an observation cannot be placed in the schedule such that it overlaps with observations which have been previously placed in the schedule. This is of course equivalent to requiring that a single telescope cannot in general service multiple observation requests simultaneously.

This latter constraint is in fact insufficient for the construction of a viable schedule. This is because sufficient time must be inserted between observations for the telescope to slew to and settle on the new target, for any previous image data to be retrieved, and for any hardware changes (e.g. filter wheel rotations) to be actioned.

The constraints described above must be satistied for every scheduled observation in order for the resulting schedule to be valid. Because of their inviolable nature, these constraints are generally referred to as *hard* constraints. Because the satisfaction of these basic constraints is required for any observation to be scheduled, these constraints are usually not explicitly listed as part of an observation request and, as such, are also examples of *implicit* constraints. All other constraints must be explicitly stated in the observing request. Examples of other constraints are the absence of the Moon in the sky during the observation, or a minimum angular separation between the Moon and the target object; a maximum zenith angle at which the target object is observed; request A must be observed no more than p hours after request B, which must be observed no less than q hours after request C. The reader is referred to Pennypacker et al. (2002) for more examples of constraints.

In addition to hard constraints, a request may also contain scheduling preferences. These are often referred to as *soft* constraints. Whilst failure to satisfy a hard constraint should result in a given request remaining unscheduled, failure to satisfy a soft constraint will not have the same effect; instead, the request will still appear in the schedule. The only impact is that the resulting schedule will not be as (qualitatively) optimal as it otherwise might have been. An example of a soft constraint is the scheduling policy for a given request, e.g. 'schedule this request to be observed as soon as possible', or, 'schedule this request to be observed as near to culmination as possible'. Certain schedule-wide preferences are also examples of soft constraints, e.g. the minimisation of telescope slew time, or the minimisation of filter wheel rotations. It should be noted that such schedule-wide preferences are generally conflicting, in the sense that constructing a schedule to maximally satisfy one soft constraint will typically satisfy other soft constraints more poorly.

3.3 Scheduling Methodology

The scheduling method adopted is a straightforward one, commonly known as 'direct heuristics', e.g. Schmidt & Strohlein (1980), Schaerf (1999). In this approach, the first phase of the scheduling process is the construction of an appropriately ordered list of observation requests for a given telescope and a given night. This list contains all of the observing requests which seek time on the telescope for that night.

Every request in the list is assigned a priority for the night, which is typically an integer value (with smaller values representing higher priorities). The list is then sorted by these priorities, such that such that the requests of highest priority appear at the top of the list.

Within each priority 'band', the requests are ordered by a 'flexibility of scheduling' (FoS) metric. This is used to ensure that, within each band, the scheduler will attempt to place to most constrained observation requests into the schedule first. The metric is calculated as the quotient of the length of the observing window for the requested target, and the time to observe the requested target. The true time required to undertake the observation is inclusive of slewing and filter wheel rotation times, both of which depend strongly on the ordering of observations in the final schedule. As such, this is a first-order approximation to the true value. Larger FoS values indicate that it is easier for the scheduler to successfully schedule this request, because a large value means that the scheduler has greater flexibility in being able to place a relatively short observation at any point within a comparatively large window.

If the schedule needs to be rebuilt, the scheduler will loop over all requests in the ordered list. It will attempt to schedule each request as near as possible to its optimal time, as determined by request preferences, whilst not violating any hard constraints. This is accomplished by identifying the optimal point in the schedule for this request, and searching outwards from that point for a candidate location. This location is checked to verify that appropriate slewing and settling times are available, and that no other hard constraints are violated. If there is a problem with this candidate location, the search continues, otherwise, the observation is inserted into the schedule at that point. This process is repeated for each request in the ordered list, until the scheduler has examined all requests, or until there is no more space left in the night's schedule. This whole process is then repeated for each telescope in the network.

The method described above is one of incremental assignment of start times (and finish times) to observation requests. As this process proceeds, each successive assignment imposes more hard constraints on the remainder of the solution. As such, more computational resources will be required (on average) to schedule request R_{n+1} than to schedule request R_n. This function is non-linear, depending strongly on the filling factor of the schedule on any particular iteration. When a schedule is nearly full (i.e. with a filling factor approaching unity), the scheduler may take considerable time to schedule a request.

When a schedule is full, the scheduler will cease attemptimg to schedule requests. Because this could happen at any iteration, care has been taken to adopt an 'anytime scheduling' approach (Pittarelli 1996). This 'anytime' approach ensures that a valid, workable schedule exists at the end of every iteration, such that a halt in scheduling activity will leave the system with a valid schedule.

It should be noted that the method outlined above produces an optimised schedule for one telescope at a time. Should a particular request be better serviced by a different telescope, it must be reallocated into that telescope's queue. The reallocation of observation requests between different telescopes in the network is performed by another subsystem, known as the 'reallocation daemon'. This daemon periodically examines current requests which remain unscheduled, and determines if they might be better serviced by another telescope. If appropriate, this daemon will reallocate the request to the new queue, whereupon it will be examined by the scheduler a few minutes later. This reallocation process can proceed differently, depending upon preferences specified in the observation request or preferences associated with a particular telescope network. The ability to set these different 'reallocation policies' makes this process quite flexible.

4 Numerical Experiments

In this section, we explore the characteristics of the scheduling engine through the use of simple, numerical trials. Only a subset of the scheduler's capabilities will be explored in this paper, in particular, these numerical trials will be limited to a single telescope, and will consist of single, independent observation requests with no explicit constraints. Experiments involving more complicated request constructs and relationships, such as multiple, associated requests serviced by different telescopes, will be addressed in future papers.

Thirty independent trials were performed. Each trial involved the injection of two hundred observation requests. These requests were for randomly-selected objects from the New General Catalogue and Index Catalogue (Sinnott 1988). Each observation consisted of thirty seconds of settling time followed by a three-minute exposure of the target object, and all were submitted to the same telescope queue. The request priorities were integer values within the range 1 to 7, inclusive, and were randomly assigned.

The telescope was located at southern mid-latitudes, near a longitude of 153° (East of Greenwich) and a latitude of $-27.5°$. All observations were scheduled for nights in the last week of June. At this location and time of year, useful night-time observations can be performed between approximately 8.5 hr and 19.0 hr UTC. The horizon of the telescope was set to a zenith angle of 55°, for all azimuths. The slewing characteristics of the telescope were set to an angular acceleration of $1° s^{-2}$ and a maximum angular rate of $3° s^{-1}$ for both axes.

Approximately forty per cent of the injected requests were immediately rejected, because these targets could not be successfully observed from this geographic location or at this time of year. The remainder of the requests (approximately 120 per trial) were available to the scheduler.

The number of requests was chosen such that the telescope was significantly over-subscribed, i.e. that there were more pending requests than could be incorporated into a valid schedule for one night. It is within this region of parameter space that the scheduler's prioritising abilities are best examined.

5 Results and Discussion

For each of the thirty sets of target objects, four different scheduling experiments were performed. Each of these experiments involved different scheduling models, as described below.

In the first model, model *A*, the priorities assigned to each of the requests were integer values in the range 1–7, inclusive. As such, there were seven priority bands for the scheduler to process. Within each band, requests were ordered using the *FoS* metric, as described in Section 3.3. The scheduler attempted to place each request in the schedule such that it was observed as early as possible. This is the model nominally used by the scheduler for its daily operations. As such, it is of interest to compare this canonical model with those below.

In model *B*, the priorities assigned to each request were ignored, effectively making a single priority band. Within this band, requests were ordered using the *FoS* metric, as in model *A* above. The scheduler attempted to place each request in the schedule such that it was observed as early as possible.

Model *C* used the assigned request priorities, so in this model there were seven priority bands for the scheduler to process. Within each band, the requests were ordered randomly. The scheduler again attempted to place each request in the schedule such that it was observed as early as possible.

The final model, model *D*, was essentially the same as model *A*, except that the scheduler attempted to place each request in the schedule such that it was observed as close as possible to transit.

Typical results from one of the numerical trials are shown graphically in Figure 1. This shows a representation of the sky as seen from the telescope. The curves represent the observable paths of the scheduled targets throughout the night, and the filled circles show the locations at which the target objects are scheduled to be observed. In the upper panel, the scheduler has attempted to observe the targets as early as possible (model *A*), in contrast, the lower panel show the corresponding schedule produced when attempting to schedule the targets as near to transit as possible (model *D*).

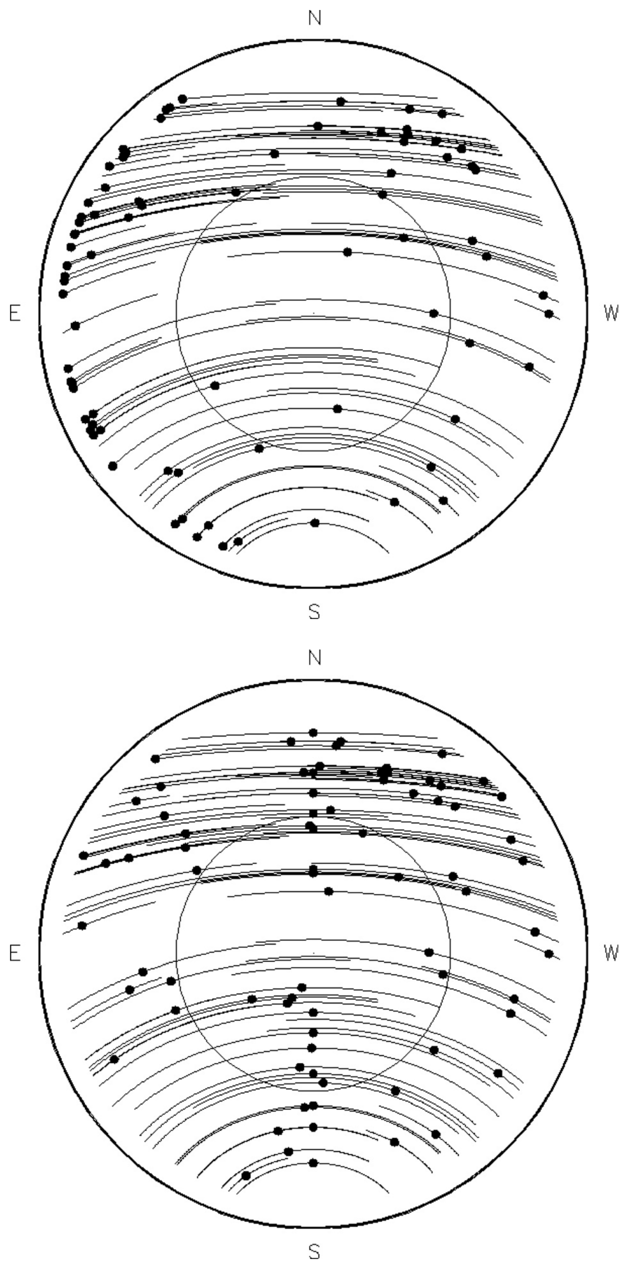

Figure 1 This figure shows the output of one of the numerical trials. The inner circle is drawn at a zenith angle of 30°, and the outer circle at a zenith angle of 60°. The arcs detail the observable paths of each of the scheduled targets across the sky, and the filled, black circles show the locations at which each target is scheduled to be observed. In the upper figure, the scheduler policy was to observe the targets as early as possible. In the lower figure, the policy was to observe the targets as near to transit as possible.

The results from all thirty trials are summarised in Table 1. This table presents the counts and the mean priorities of those queued requests which were successfully scheduled, and those which remained unscheduled. These values are averages over all of the numerical trials.

Table 1. Results of the scheduling trials

Model	Priority banding	Within-band ordering[a]	Scheduling preference	Scheduled count	Unscheduled count	Scheduled priority	Unscheduled priority
A	Multiple	*FoS*	Earliest	80 ± 5	41 ± 4	3.4 ± 0.2	5.1 ± 0.2
B	Single	*FoS*	Earliest	96 ± 4	25 ± 6	4.0 ± 0.2	4.0 ± 0.3
C	Multiple	Random	Earliest	73 ± 4	47 ± 4	3.4 ± 0.2	4.9 ± 0.2
D	Multiple	*FoS*	Transit	76 ± 5	42 ± 4	3.4 ± 0.2	4.9 ± 0.2

[a] The term *FoS* refers to the 'flexibility of scheduling' metric discussed in Section 3.3.

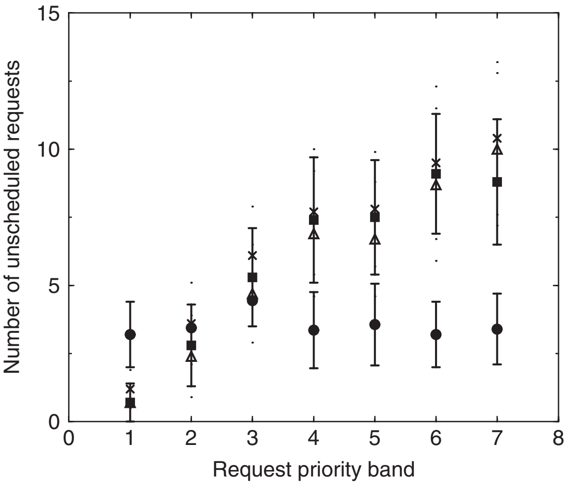

Figure 2 This figure shows the mean number of requests, as a function of request priority, which remained unscheduled from each model. As described in Section 5, the triangles show the data from model *A*, the filled circles the data from model *B*, the crosses the data from model *C*, and the filled squares the data from model *D*.

Additionally, the number of requests remaining unscheduled in each model, along with the corresponding request priorities, are plotted in Figure 2. These results can be used to quantify the effect of the different scheduling models on the overall efficiency of the engine.

5.1 Priority Banding

Comparing models *A* and *B* shows the effect of scheduling a single band of observation requests, as opposed to seven priority bands.

It is clear from Table 1 that using the single-band model results in an increased number of scheduled requests. However, because the single-band model does not preference higher-priority requests over lower-priority requests, the resulting schedule may not be optimal from this perspective. Nevertheless, this difference in the number of scheduled requests is the largest resulting from any single change in the scheduling parameters.

Further investigation revealed that the reduced number of scheduled requests in the multi-band model was due to the latter model departing from the strict *FoS* ordering of the single-band model. In other words, the multi-band model will have many observation requests with smaller *FoS* values being examined by the scheduler after requests with larger *FoS* values. An increased number of these smaller-valued requests will not be able to be successfully scheduled as a result. In the single-band approach of model *B*, the *FoS* values of the candidate requests are guaranteed to monotonically increase as the scheduler considers each in turn. As such, more observations can (in general) be successfully scheduled in the single-band case.

The magnitude of this effect is surprising. The author is currently experimenting with a 'reduced multiband' approach, in which the number of priority bands is limited to a small number (no more than three or four). Whilst this is not investigated herein, it is thought that this approach will preserve the scientific benefits of preferentially scheduling higher-priority requests, whilst also retaining some of the efficiencies evident in the single-band approach.

5.2 'Flexibility of Scheduling' Metric

Comparing models *A* and *C* shows the effect of using the *FoS* metric to order the observation requests within each of the seven priority bands, as opposed to random ordering within the bands.

Utilising the *FoS* metric for ordering has already been shown to be of benefit in increasing the number of requests which are successfully scheduled (see Section 5.1).

In comparing models *A* and *C* in Table 1, it can again be seen that the use of the *FoS* ordering increases the number of requests in a schedule. This is a smaller effect than that evidenced in Section 5.1 above, although it should be noted that the multiband nature of model *A* (and the consequent non-monotonic sequence of *FoS* values) will have reduced the effectiveness of the *FoS* ordering.

The origin of the difference in the performance of the scheduler between models *A* and *C* is basically the same as that identified in Section 5.1 above. The random ordering of *C* means that more requests with smaller *FoS* values will be examined following requests with larger *FoS* values. A larger number of these smaller-valued requests will not be able to be successfully scheduled as a result.

5.3 Early vs Transit

Models *A* and *D* differ in their scheduling preference, with model *A* specifying 'early' and model *D* 'transit'. Specifying 'early' means that the scheduler will attempt to schedule all requests such that they will be observed as early as possible. In contrast, 'transit' will result in a schedule in which all of the requests will be observed as near to transit as possible.

Table 2. Mean hour angles for models *A* and *D*

Model	Policy	*HA* (hr)	$|HA|$ (hr)
A	Early	-1.48 ± 0.30	2.70 ± 0.16
D	Transit	$+0.01 \pm 0.21$	1.39 ± 0.26

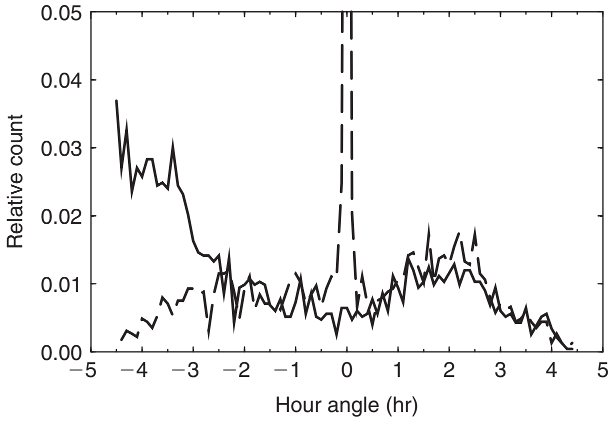

Figure 3 This figure shows the distribution of hour angles at which target objects are scheduled to be observed. The solid line shows the results for the 'early' scheduling policy (model *A*), and the dashed graph shows data from the corresponding schedules when run with the 'transit' policy (model *D*). In both cases, hour angle values were divided into bins of width 0.1 hr. The data for each graph have been normalised such that the integrated counts sum to unity. Although not shown with this vertical scale, the central ($HA = 0.0$ hr) bin of the 'transit' graph has a value of 0.255. See Section 5.3 for further discussion of these data.

First, it is of interest to quantify the effect of this preference difference on the hour angles at which requests are scheduled to be observed. Table 2 shows the mean of the absolute hour angle values to be approximately 2.7 hr in the 'early' case and 1.4 hr in the 'transit' case. These are averages over all of the numerical trials. Note that the absolute hour angles are used here, because it is how close to the meridian an observation is scheduled that is of interest – whether an observation is scheduled before or after transit is not relevant.

Figure 3 shows the distributions of hour angles for the 'early' and 'transit' models. The 'early' graph shows a large number of objects being scheduled near the eastern horizon of the telescope, with approximately 25 per cent of all observations scheduled for within one hour of rising, and 50 per cent within 2.5 hr of rising. In contrast, the 'transit' graph shows few observations scheduled near the eastern or western horizon, instead exhibiting a large peak near transit. The centre bin of this graph (representing $HA = 0.0$ hr) accounts for approximately 25 per cent of all scheduled observations.

Interestingly, both graphs in Figure 3 exhibit a broad peak between hour angles of approximately +1.0 hr and +2.5 hr. This feature is produced by the Virgo cluster, centred near (RA, Dec) of (12^h30^m, $+10°$), which is in the western sky, and hence at positive hour angles, in the

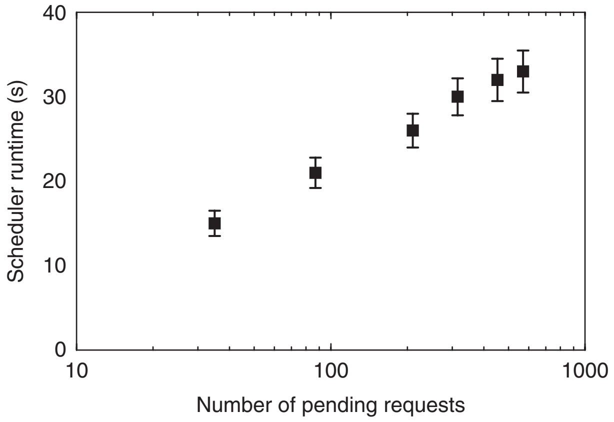

Figure 4 This figure plots the time taken to generate a schedule as a function of the number of requests in the queue. These data are discussed in Section 5.4.

early evening for the location and date of these numerical experiments.

It can be seen in Table 1 that the 'transit' policy results in fewer requests being scheduled than the 'early' policy. This is the result of fragmentation of the schedule introduced by the 'transit' policy. When using the 'early' policy, requests tend to be placed into the schedule so as to form tightly-packed blocks, with little or no free time between requests. This is a natural consequence of the 'observe as early in the night as possible' approach. In contrast, the 'transit' policy leaves slightly longer gaps of free time between the requests. In this sense, the final schedule is not as tightly packed, and it is this slight fragmentation which is responsible for the reduction in the number of requests which are successfully scheduled.

5.4 Time Taken to Generate a Schedule

In addition to the experiments described in the preceeding Sections, another series of numerical trials was performed in order to characterise the amount of time required to construct a schedule. In these experiments, various numbers of requests were injected for scheduling, ranging from approximately 30 to 600 requests. The time required for the scheduling engine to construct a schedule from these pools of requests was measured, and these results are plotted in Figure 4.

These data points show a logarithmic progression, and are well fitted by the equation

$$t = (6.6 \pm 0.2) \ln N - (8.4 \pm 1.1) \qquad (1)$$

where t is the time required to generate a schedule (in seconds) and N is the number of requests processed by the engine.

Values of t are likely to depart from this logarithmic curve for large values of N because a certain level of pre-processing is required for each request passed to the scheduler, which should scale linearly in N rather than $\ln N$. This pre-processing is not very computationally intensive, so departures from $\ln N$ are likely to be significant only for $N \gg 10^3$.

It is worth noting that these experiments were performed with a 900-MHz Intel Celeron processor. Running similar experiments on a high-end system would be likely to reduce values of *t* by a factor of several.

6 Conclusion

This paper has described a scheduling engine which is used to construct schedules for a network of small-aperture, automated telescopes. For the purposes detailed herein, this scheduler is very effective, producing consistent, well-optimised schedules quickly and reliably. It reacts rapidly to changing circumstances or unanticipated events, regenerating schedules appropriate to the new circumstances.

The approach of generating each telescope's schedule individually may not lead to the most optimised set of schedules across the (multi-telescope) network. With this in mind, a new scheduling engine is currently under development. This new scheduler will generate schedules in parallel for all telescopes in the network. It is expected that this will result in greater level of satisfaction of request preferences.

7 Availability

The system described herein is being made available to the general public, such that they can add their telescopes to the network and submit requests. The system may be accessed at the URI of `http://www.apta.net.au/`.

References

Adelman, S. J., Dukes, R. J. & Adelman, C. J., 1992, ASP Conf. Ser. 28, Automated Telescopes for Photometry and Imaging (San Francisco: ASP)

Allen, J. R., 1999, 'Driving by the Rear-View Mirror: Managing a Network with Cricket', in Proceedings of the First Conference on Network Administration (Santa Clara: Advanced Computing Systems Association), `http://www.usenix.org/events/neta99/allen.html`

Barták, R., 1999, in Proceedings of Week of Doctoral Students (WDS99), Part IV (Prague: MatFyzPress), 555

Barták, R., 2002, Neural Network World (Prague: Institute of Computer Science), 12(5), 421

Denny, R., 2005, ACP Observatory Control Software, `http://acp.dc3.com/i`

Denny, R., 2004, in Proceedings of the 23rd IAPPP Western Wing Conference: Symposium on Telescope Science (California: IAPPP Western Wing, Inc.), in press

Dierks, T. & Allen, C., 1999, The TLS Protocol Version 1.0, RFC 2246, `http://www.ietf.org/rfc/rfc2246.txt`

Drummond, M., Bresina, J., Edgington, W., Swanson, K., Henry, G. & Drascher, E., 1995, in ASP Conf. Ser. 79, Robotic Telescopes: Current Capabilities, Present Developments, and Future Prospects for Automated Astronomy, Eds. Henry, G. W. & Eaton, J. A. (San Francisco: ASP), 101

Edgington, W., Drummond, M., Bresina, J., Henry, G. & Drascher, E., 1996, in ASP Conf. Ser. 87, New Observing Modes for the Next Century, Eds. Boroson, T., Davies, J. & Robson, I. (San Francisco: ASP), 151

Fielding, R. et al., 1999, Hypertext Transfer Protocol – HTTP/1.1, RFC 2616, `http://www.ietf.org/rfc/rfc2616.txt`

Filippenko, A. V., 1992, ASP Conf. Ser. 34, Robotic Telescopes in the 1990s (San Francisco: ASP)

Henry, G. W. & Eaton, J. A., 1995, ASP Conf. Ser. 79, Robotic Telescopes: Current Capabilities, Present Developments, and Future Prospects for Automated Astronomy (San Francisco: ASP)

Hessman, F. V., 2004, Remote Telescope Markup Language 3.0, `http://alpha.uni-sw.gwdg.de/~hessman/RTML/`

Oswalt, T. D., 2003, The Future of Small Telescopes in the New Millennium (Dordrecht: Kluwer)

Pennypacker, C. et al., 2002, A&A, 395, 727 doi:10.1051/0004-6361:20021318

Pennypacker, C. et al., 2003, in The Future of Small Telescopes in the New Millennium. Volume I – Perceptions, Productivities, and Policies, Ed. Oswalt, T. D. (Dordrecht: Kluwer), 97

Pittarelli, M., 1996, SIGART Bulletin Special Issue on Anytime Algorithms and Deliberation Scheduling (New York: ACP), 7(2)

Richmond, M. W., Treffers, R. R. & Filippenko, A. V., 1993, PASP, 105, 1164 doi:10.1086/133294

Schaerf, A., 1999, Artificial Intelligence Review (Dordrecht: Kluwer), 13, 87

Schmidt, G. & Strohlein, T., 1980, The Computer Journal (Oxford: Oxford University Press), 23(4), 307 doi:10.1093/COMJNL/23.4.307

Sinnott, R. W., 1988, NGC 2000.0, The Complete New General Catalogue and Index Catalogue of Nebulae and Star Clusters (Cambridge: Sky Publishing Corporation and Cambridge University Press)

Steele, I. A. & Carter, D., 1997, Proc. SPIE 3112, Telescope Control Systems II, Ed. Lewis, H. (Washington: Bellingham), 222