ADAPTIVE OPTIMAL TELESCOPE SCHEDULING

STEPHEN NICHOLAS FRASER

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy

May 1, 2012

Abstract

Astronomical telescopes operate in a dynamic and uncertain operational environment. Time is often oversubscribed with programs competing for available slots and the best observing conditions. In order to achieve a high scientific return a scheduler must be able to create a globally optimal observing schedule.

Using data collected from external sources and embedded instrumentation I investigate and characterize the scheduler's operating environment. I investigate metrics for characterizing the value of schedules both locally (at the decision point) and over longer horizons. Using this information an extensible software architecture is designed incorporating a simulation framework to allow a variety of schedulers and environmental scenarios to be constructed.

Experiments are performed using the scheduler component architecture and simulation framework to determine the effects on schedule quality of environmental stability, disruptive events and reliability of prediction under a range of load conditions.

Investigations into the effects of choice of selection scoring metric on the final quality of schedules showed that contrary to an initial opinion that it should be feasible to find a single candidate quality metric with which to evaluate schedule quality, there is in fact no clear candidate as an absolute quality metric. The choice of weighting of the selection metrics determines the character of the schedule but only the dominant scoring metric has any real effect while the other metrics simply manifest themselves as noise.

It was found that where conditions are relatively stable there is an advantage to using longer look-ahead horizons but that where conditions are unstable or disruptive a basic despatch scheduler achieves results as good or better. A look-ahead scheduler with a horizon of 4 hours can give an improvement of upto 23% over a simple despatch scheduler when seeing remains stable over a period of upto 6 hours.

Experiments performed to determine the effect of chosing a suitable look-ahead horizon based on the perceived stability of environmental conditions have shown that when determining the length of this horizon, if we either over or under-

estimate the duration of stable conditions we obtain poorer results. When the stability length is severely under-estimated a loss of between 5% and 17% in quality was found to occur. The optimum situation appears to be when the horizon is of order of or slightly less than the actual stability length. Where the stability length is over-estimated by a factor 2, a loss of 5% to 10% can occur.

Contents

List of Figures

List of Tables

Glossary

ACL	Aggregate Components Layer - part of the SCA, contains summaries of base models	125
ACO	Ant Colony Optimization	57
AIS	Artificial Immune System	57
BCS	Boltwood Cloud Sensor - a component attached to the WMS for measuring cloud temperture	114
BDS	Basic despatch Scheduler - a simple scheduler	149
CAMC	Carlsberg Meridian Circle - a specialized telescope at the ORM	113
CBR	Case Based Reasoning - an AI technique	65
CP	Cardinal Pointing - a restricted operating regime for the rotator axis	136
CTL	Compuational Tools Layer - part of SCA, provides processing capabilities	126
ECL	Executive Components Layer - part of the SCA, contains the external applications which invoke the scheduler	140
EFR	Expected Future Reward - a method for discounting the value of potentially uncertain future rewards	150
ELAS	Enhanced Look-ahead Scheduler - a simple look-ahead scheduler based on QLAS but with a Sequestration Policy	151

FCL	Fundamental Components Layer - part of the SCA, contains models for accessing the ODB	124
FITS	Flexible Image Transport System - an astronomical data file specification	105
GA	Genetic Algorithm	42
GTC	Grand Telescopio Canarias - a telescope at the ORM	104
ICL	Interface Components Layer - part of the SCA, contains interfaces used by the executive components	138
ISM	Instrument Synopsis Model	131
JKT	Jacobus Kaptyn Telescope - a telescope at the ORM	114
LT	The Liverpool Telescope	28
MUF	Minimum Use Fraction - a TAG assigned target for program completion	81
MDP	Markov Decision Process	63
MLP	Multi Layer Perceptron - a simple type of neural network	63
NOT	Nordic Optical Telescope - a telescope at the ORM	104
ODB	Phase 2 Observation Database	28

ORM	Observatorio del Roche do los Muchachos	28
OSS	Observation Scheduling System	28
PCL	Prediction Components Layer - part of SCA, provides predictive models	129
PCM	Problem Complexity Metric, C metrics are used to measure the complexity of a given scheduling problem in terms of Phase 2 structure	69
PSO	Particle Swarm Optimization	62
QA	Quantum Annealing	41
QLAS	Quantum Look-ahead Scheduler - a simple look-ahead scheduler using a random sequence generator	151
RCS	Robotic Control System	28
RL	Reinforcement Learning	62
SA	Simulated Annealing	40
SCA	Scheduler Component Architecture	123
SEC	Simulation Experiment Controller - a control application specific to a simulation experiment	143
SECL	Scheduling Engine Components Layer - part of the SCA	134
SFL	Simulation Framework Components Layer - part of the SCA, provides models used by simulators	132
SNR	Signal to Noise ratio	84
SQM	Schedule Quality Metric, Q metrics provide a measure of scheduler performance	69

TSG	Time Signal Generator - a SCA component which synchronizes the scheduler with its simulation controller	132
TSM	Telescope Synopsis Model	131
TSP	Travelling Salesman Problem	60
WMS	Weather Monitoring System	90

List of Symbols

a	The set of account synopses apertaining to the current group	122
C_A	Actual contention as measured during one or more runs of the scheduler	70
C_C	Contention profile	70
C_D	Demand profile	71
C_L	Schedule load	73
C_{CL}	Critical load	73
C_{UL}	Urgency weighted load	73
$\Delta_{90}(C, B)$	Fractional improvement in quality of C over B	153
E_{FA}	Fixed environment model with average seeing ($0.8'' < s < 1.3''$)	129
E_{FP}	Fixed environment model with poor seeing ($s >$ 1.3”)	129
E_{FX}	Fixed environment model with excellent seeing ($s < 0.8$”)	129
$e(t)$	The current (at time t) environmental conditions	122
$\Phi(g, t, e, a, h)$	Feasibility model, determines whether a particular group is feasible subject to its constraints and under specified conditions	126
f_h	Scoring metric based on target airmass	135
f_h	Scoring metric based on target elevation	135
f_{moon}	Scoring metric based on matching requested and actual sky brightness	136

f_{oa}	Scoring metric derived from Q_{OA}	136
f_{oh}	Scoring metric derived from Q_{OH}	136
f_{phot}	Scoring metric based on matching requested and actual extinction	136
f_p	Scoring metric derived from Q_{PX}	136
f_{rn}	Scoring metric derived from Q_{RN}	137
f_{see}	Scoring metric based on matching requested and actual seeing	136
f_{slew}	Scoring metric based on reducing slewing overheads	136
f_{td}	Scoring metric derived from Q_{TD}	137
f_{trans}	Scoring metric derived from Q_{OH}	135
f_{yt}	Scoring metric derived from Q_{YT}	137
g	A composite group (the group currently under consideration)	121
h	Synopsis of the execution history of the current group	122
Q_{BA}	Airmass benefit quality metric	78
Q_{LM}	Lunar condition matching quality metric	81
Q_{MW}	Monitor Window quality metric	78
Q_{OA}	Optimal airmass quality metric	78
Q_{OH}	Optimal height quality metric	77
Q_{PC}	Project Completion quality metric	80
Q_{PX}	Execution Priority quality metric	79
Q_{RN}	Remaining Nights quality metric	80
Q_{SM}	Seeing condition matching quality metric	81

Q_{SU}	Score Utility quality metric	81
Q_{TD}	Target Demand quality metric	79
Q_{XT}	Execution Time quality metric	79
Q_{YT}	Yield Tracking quality metric	81
$S(g, t, e, a, h)$	Score for a group g at time t under specified account balance, history and environment conditions	135
τ_B	Length of time conditions remain *bad*	98
τ_G	Length of time conditions remain *good*	98
t	The current (real or simulation) time	121
$V_e(g, t)$	Enterprise utility for executing group g at time t	74
$V_u(g, t)$	User preference utility for executing group g at time t	75
Δ_W	Fraction of night during which weather conditions are **not** acceptable for observing	101
$X(g)$	Execution time of a group	71
ζ	Scheduler candidate selection model	137
ζ_{Best}	Best score selection model	186
ζ_{FR}	Fixed rank selection model	187
ζ_{RS}	Rank scaled selection model	187

Part I

Background

1 Introduction

1.1 Background

The Liverpool Telescope (LT),(Steele, 2004) is a fully robotic 2 meter astronomical telescope situated at the Observatorio del Roche de los Muchachos (ORM) on the island of La Palma in the Canary Islands ($28°45N$, $17°52W$) at an elevation of 2400m. It operates without supervision, controlled by an integrated suite of software systems (Steele and Carter, 1997).

The telescope operates in an uncertain environment:- (*i*) availability is influenced by weather conditions (it does not open when there is rain, high humidity or extreme or gusting wind), (*ii*) at times telescope operations may be *taken over* by external agents (Mottram, 2006; Allan et al., 2004) to investigate *targets-of-opportunity*, (*iii*) essential engineering work must be scheduled at various times, (*iv*) there are the inevitable and unexpected downtime due to software and hardware faults, (*v*) atmospheric conditions which constrain the choice of available observation can change over short timescales, (*vi*) observation requests can arrive in the database or be removed or modified at any time.

The telescope's Robotic Control System (RCS) (Fraser and Steele, 2002), is responsible for deciding on the mode of operation and for protecting the telescope. It controls the telescope and its instruments and monitors environmental information from a number of lower level subsystems. The observation program followed by the RCS is determined by the Observation Planning and Scheduling System (OSS) based on information in the Phase 2 Observation Database (ODB).

The ODB contains details of observations specified by astronomers who have been allocated time by one of several Telescope Allocation Committees/Groups (TAC/TAG)[1]. The term *group* (Fig. 1.1) is used to describe the observation specifications. These

[1]The 2 terms are synonymous, TAG is more usual in UK, TAC in USA

contain details of what and how to observe (*sequence*), when to observe (*timing constraints*) and under what conditions (*observing constraints*).

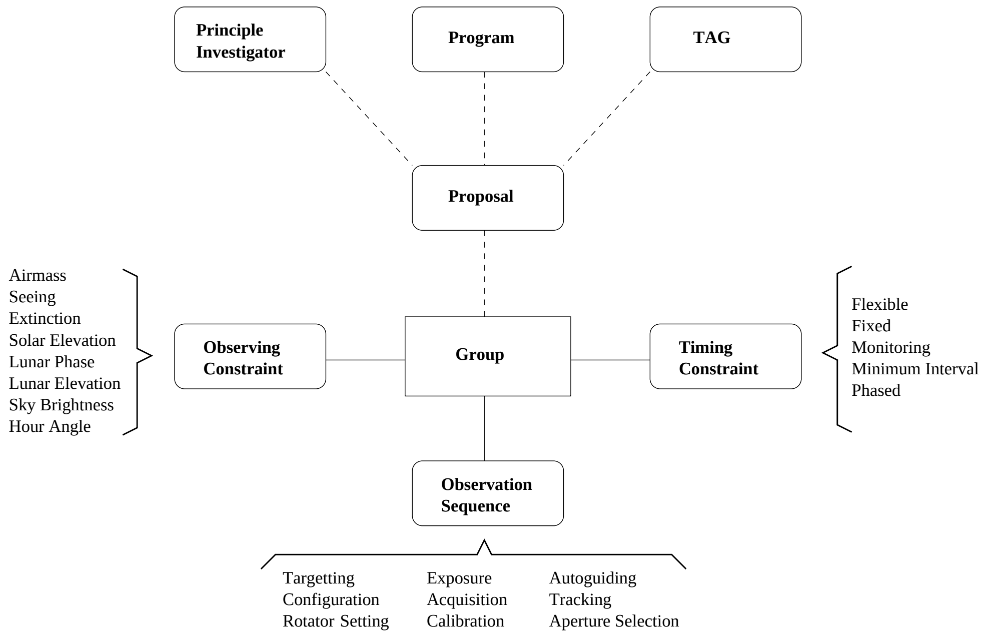


Figure 1.1: Structure of the Phase2 Observing Database (ODB). A *group* contains all the information required to schedule and execute a series of observations. The *observation sequence* contains details of the operations to be performed (target, exposure, instrument selection). *Timing constraints* determine when the group may be executed. *Observing constraints* determine under which atmospheric and other conditions the group is feasible. Groups are contained within a *proposal* containing details of the science program and time allocations from its sponsoring TAG. Each proposal has a *Principle Investigator* (and possibly *co-inverstigators*) who are responsible for entering the observation details.

The original OSS deployed at the telescope (OSS-A; Fraser, 2004), was developed based on the assumption that a despatching scheduler would be best at matching the varying conditions on site with the timing and observing constraints of the available groups. A despatch scheduler works by scanning the set of available groups. Each group which is determined to be feasible (with respect to the current conditions) is assigned a score. Typically, the group with the highest score at that moment in time is then selected for execution.

1.2 Planning and scheduling

The terms planning and scheduling are often confused and indeed used interchangeably. They are however basically different aspects of the same process. Planning is typically considered to be a long term activity taking a global view of an enterprise. It may set targets or goals to be achieved by the enterprise. These may be somewhat imprecise or abstract - we are not deciding *what* to do and *when* so much as what *sort* of things we want to do and *how well* we want to achieve them. Scheduling on the other hand deals with the rather more precise objective of assigning actual resources to accomplish specific tasks at specific times.

The planning *horizon* is the period over which a plan is made or is expected to be of some utility in achieving its set of goals. In practice we can usefully subdivide planning into several different levels characterized by their horizons (Hax and Meal, 1973). The subdivisions are by no means set in stone and more or fewer horizons may be more suited to specific enterprises. As we move down the hierarchy, typically, the precision of the objectives becomes more focused and the horizon time span decreases (Chien et al., 2000).

Strategic planning involves the creation of plans to cover very long periods of time. In the current context this could be a semester of telescope time, and more generally could at its most extreme represent the entire lifetime of an enterprise.

Tactical plans are designed to cover medium periods e.g. a lunar cycle or half cycle on a telescope, weekly or monthly production targets in a factory.

Operational plans [2] are short term plans designed to allow an enterprise to function on a day to day basis. Typically they cover the *standard* work-period of the enterprise e.g. a night's observing at a telescope, a shift at a factory.

[2]may also be referred to as *Mission planning*

Scheduling Takes place over a short period and deals with the specific assignment of resources to tasks. This could range from determination of just the next observation up to the specification of the sequence of observations over the next few hours.

In this thesis I shall consider only the problem of scheduling. Though levels of planning may be incorporated into a final operationally deployed system they are likely to manifest themselves through adjustment of the relative weightings of the quality metrics used by the schedulers over periods longer than a night.

1.3 Motivation

The original scheduler system (OSS-A) has been in operation now for 7 years since the telescope went on sky (2004) and has performed adequately, however a number of criticisms can be made concerning this implementation.

- There is no *commonly agreed* mechanism for measuring what represents a *good* schedule (but see (Steele and Carter, 1997)). In order to compare the results of different scheduling implementations we need some means of determining the value or reward associated with a schedule. This can be considered from 2 points of view. For the user, the highest reward is likely to be achieved if his observations are made regularly, at the most favourable times with respect to the timing constraints and under the most favourable environmental conditions. For the enterprise, the highest reward might be achieved if the observations selected are most *valuable* in terms of TAG assigned priority and may require some degree of program completeness to be satisfied.
- We do not have a detailed understanding of how conditions change on site and how this might affect planning and scheduling. At present no consideration of environmental change is considered in the planning and scheduling process.

Similarly there is no real understanding of the effects on scheduling of the potentially rapidly changing content (volatility) of the ODB.

- Despatch schedulers have been described as *myopic* (Cicirello and Smith, 2001), in that they select the *best group* to execute at a point in time without reference to what might lead to an overall good schedule. There is no consideration of whether a better sequence might be generated by looking ahead a few steps. This is an example of *local* rather than *global* optimization.

It was therefore decided to conduct a series of experiments using different scheduler paradigms in order to determine how these would perform under a range of environmental and load scenarios.

1.4 Plan

The thesis is divided into 2 parts. In the first part, information is gathered to measure and characterize the operating environment and then used to design a scheduling architecture.

1. In Sect. 3 an investigation is made into the subject of metrics and reward. Two types of metric are considered. Complexity (C) metrics are applied to the measurement of ODB characteristics. These indicate how *difficult* a particular scheduling problem might be. Quality (Q) metrics are used to measure the reward or value of a generated schedule and allow a comparision between the success of different scheduler implementations or between different problem sets.

2. Software embedded within the current robotic control system and scheduler is used to collect information relating to the characteristics of the telescope's operating environment. This information is analysed in Sect. 4 to determine whether it will be possible to incorporate predictions of environmental conditions into decision making. Mitigation of the effects of changing conditions are considered by looking at the concept of discounted future reward.

3. Information gathered in the 2 preceding stages is used to design an architecture with which to build a variety of scheduler implementations and incorporating components with which to construct simulation environments to test these schedulers. Heuristic scoring (f) metrics are employed by the scheduler and planner implementations to perform search and optimization processes. The scheduler architecture is described in Sect. 5 with further details of the simulation components in Sect. 6.

In the second part of the thesis a number of simulation experiments are performed using the framework described in Sect. 5

Two different types of scheduler are considered. The *despatch* scheduler, as currently operationally deployed works by considering which of the various feasible candidates achieves the best score under a particular scoring model at the current time under the current conditions. The *look-ahead* scheduler examines a number of feasible sequences of observations starting from the current time and looking forward by a single horizon length. It compares each sequence selecting the one which maximizes the potential reward.

1. In Sect. 7 a human scheduler is pitted against a simple despatch scheduler to act as a shakedown for the framework and to provide a baseline against which to test further schedulers.

2. The effect of varying the weights of the various scoring metrics on schedule quality is tested in Sect. 8.

3. An investigation into the scale and range of Phase 2 loading is made in Sect. 9. Schedulers are tested against generated Phase 2 models of differing complexity to determine how this affects scheduling performance.

4. Experiments are performed in Sect. 10 to investigate the effect of variable environmental stability on schedulers and to determine which schedulers if any perform best under differing stability regimes.

5. Unpredicted disruptive events can take place during a night's observing. An investigation into the effects of these disruptive events on schedule quality is made in Sect. 11.

6. When the content of the Phase 2 ODB is changed during the course of the observing night there is likely to be some effect on scheduling quality. A detailed study of the effects of Phase 2 volatility are investigated in Sect. 12 in order to determine the extent of this effect.

7. Look-ahead schedulers must decide at some point on how long a forward horizon they should use. In Sect. 13 simulations are performed to investigate how the reliability of determination of such horizon lengths based on environmental stability criteria affects this class of scheduler.

2 Review of current research

There is a vast literature on the subject of scheduling covering both generic problems and highly domain-specific problem areas. In this section I present a brief introduction to some general techniques and review a number of specific case-studies.

The constructive and iterative schedule building paradigms are introduced in Sect. 2.1 along with details of a number of search techniques used to speed up the scheduling process. A number of individual case studies are described in more detail in Sect. 2.2. Flexible and reactive scheduling methods are described in Sect. 2.3. In Sect. 2.4 a number of agent-based distributed scheduling systems are discussed. A number of biological and evolutionary based scheduling paradigms is described in Sect. 2.5. The use of artificial intelligence (AI) techniques including adaptive learning methods are described in Sect. 2.6.

2.1 General techniques

2.1.1 Constructive techniques

The constructive approach to generating schedules starts from an empty schedule, progressively selecting tasks to assign to vacant time slots, gradually building up the schedule. Effectively it generates a path through the search space. As the search progresses, dead-end points may be reached where some task becomes nonassignable. The search must then backtrack, unwinding previous assignments and attempting to reassign tasks in order to avoid these dead-ends. An uninformed search can be very inefficient, however, search performance can be improved by using domain knowledge to derive heuristics (rules of thumb) via some of the following techniques:-

Variable ordering heuristics determine the order in which variables [3] are selected for assignment. In scheduling this corresponds to the order in which tasks are

[3] In this context, *variable* refers to some resource which is to be allocated. In this thesis I will be concerned with a single variable *time*.

selected for time-slot assignment. The *minimum remaining value* (MRV) heuristic (Bitner and Reingold, 1975) always selects the variable with the fewest legal assigments to place next, causing the search to fail fast and allows rapid pruning of the search tree. Sadeh (1991) describes a number of variable ordering heuristics used in the MicroBOSS scheduler. These are split into *fixed variable order heuristics* where the order is predetermined at the start of the search and *dynamic variable order heuristics* where the order is revised each cycle. The *MinimumWidth* heuristic selects the variable with the fewest arcs, i.e. constraint associations with other variables. Their *operation resource reliance* (ORR) variable order heuristic selects the task which relies most on the most contended resource or time period. In the *MinConflicts* heuristic (Minton et al., 1992), at each point in the search a variable is selected which is in conflict and its value is adjusted until it is no longer in conflict.

Value order heuristics determine how the value is chosen to assign to a selected variable. Sadeh (1991) describes a *filtered survivable schedules* (FSS) heuristic which assigns to an operation the time-slot which is likely to be compatible with the largest number of survivable schedules or chances of surviving competition with other operations for possession of a resource. The technique employed detects *cliques*, areas of the constraint graph with the tightest constraints. They also describe a mechanism to allow the scheduler to switch to a simpler (cheaper) value order heuristics when contention drops below a threshold.

Constraint propagation A more general improvement can be made using constraint propagation techniques. Here the implications of a constraint on a variable are tested against the constraints on other connected variables. For arc-consistency there must be a consistent assignment of variables to X for every valid assigment of connected variable Y, if not we must delete values from one or other domain. The effects are then propagated to neighboring arcs. The name stems from the way new constraints are inferred and added to the constraints set. k-consistency takes this further by insisting that for every $k-1$ assigned variables a consistent value can be assigned to any k^{th} variable. Conflict analysis though costly (typically $O(e^n)$) (Bitner and Reingold, 1975) can reduce backtracking by pruning the search tree. There is a trade-off in the time taken to perform the consistency checking and the reduction in problem size generated.

Johnston and Miller (1994) use node ($k = 1$), arc ($k = 2$)and path ($k = 3$) consistency to prune the search space for scheduling observations with the Hubble Space Telescope (HST). An example of arc-consistency between binary-constrained variables is given where two observations A and B with a precedence constraint B ***after*** A by at least Δt with each observation having unit duration (for simplicity) and restricted to the interval $[t_A, t_B]$ then the sub-interval $[t_A, t_A + \Delta t + 1]$ is excluded from the domain for B and the subinterval $[t_B - \Delta t - 1, t_B]$ is excluded from the domain for A. The trade-off in time spent consistency checking against problem reduction is handled in SPIKE by enforcing a strict time-limit for this procedure.

Deadend recovery heuristics The occurance of deadends resulting in the need to perform backtracking indicates that the chosen variable and value ordering heuristics and consistency enforcing techniques are insufficent to cope with the problem in hand. Further, a consequence of re-application of these same heuristics on backtracking is that the same deadends may be encountered repeatedly, a thrashing effect. Recovery heuristics are designed to allow for more intelligent choice on how to backtrack. Sadeh et al. (1994) describes several general techniques for improving backtracking search using the *partial conflict set* of the deadend (i.e. the set of activities which have blocked progress of the search at that point and which may have been involved in previous deadends).

- *Dynamic Consistency Enforcement* (DCE) keeps a history of backtracking events to identify resource critical subproblems and unwinds assignments until a consistent state is reached.
- *Learning Order from Failure* (LOFF) technique attempts to adjust the vari-

able ordering heuristic on encountering a deadend by unwinding to a consistent state then overriding the default heuristic to assign the activities in the partial conflict set before those which would otherwise have been chosen.

- *Incomplete Backjumping Heuristic* (IBH) uses texture based measures (Beck et al., 1997) to identify assignments which are estimated to lead to more global solutions so that when a deadend is detected unwinds to this critical assignment and tries alternatives - e.g. use second best assignment rather than best.

A disadvantage of the constructive approach is its *offline* nature. A schedule generated by any offline technique will generally be impossible to follow in dynamic environments for any length of time. During execution any breaks are likely to cause the whole schedule from that point onwards to have to be regenerated, perhaps frequently, at some cost in time and computing resources.

2.1.2 Iterative repair

Iterative repair based scheduling starts with an initial, probably sub-optimal and possibly infeasible schedule solution. It then attempts to repair this by iteratively swapping assignments using local search techniques. The repair operations may involve *retraction heuristics* for determining conflicting or over-subscribed tasks to remove, followed by the use of *replacement heuristics* to determine which new assignments to make.

Repair may occur as part of a search process to find an optimal schedule from a preliminary first guess. It may also be neccessary to repair in a reactive context to fix an already executing schedule. This might have been disrupted by unexpected changes to the environment or goals or used opportunistically to take advantage of such changes to increase the global value. A task might finish earlier than expected which could provide an opportunity to bring another task forward.

Several such retraction and task-swapping heuristics are described by Kramer and Smith (2004) and later in Sect. 2.2.5 of this review.

2.1.3 Search techniques

Before embarking on a review of scheduling methodologies we will briefly examine some of the available search techniques which have been applied to scheduling.

We must first distinguish between local and systematic search. Local search is characterized by operating around a small neighbourhood of the search space. It is generally memoryless or has a very short memory.

Systematic search however generally contains some sort of memory of previous search areas, operates on a larger scale and leads to the concept of global optimization.

Local search Typically this proceeds by making small moves around the *current* neighbourhood in the search space. Various types of local search are characterized by the mechanism for selecting the *move* operation. These heuristics or *rules of thumb* are guided by specialized domain knowledge and often a degree of experimentation and tuning is required to achieve acceptable performance.

Greedy search also known as hill-climbing (or gradient descent depending on whether the aim is to maximize or minimize the objective), is a classic local search technique. The move operation steps one unit in the direction of highest upward (or downwards) change in objective function, climbing (or descending) the hill following the steepest gradient. The search continues until the peak (or valley) has been found as quickly as possible. In smooth search spaces this can be a very effective technique. Where the search space has a *rough* texture (Beck et al., 1997; Fox et al., 1989), this search mechanism can become trapped at *local* maxima (or minima) with no possibility of escape to allow the search to explore other parts of the space. If the roughness of a search space is well understood, and this is often difficult as search spaces are frequently multi-dimensional and not easily visualized, then a local strategy may prove adequate. Often however, the *texture* of the space may well not be known in advance, so the effectiveness of local search heuristics can be limited. In an attempt to allow search mechanisms to escape these localization effects a number of techniques have been developed.

Search with noise / stochastic local search In a typical scenario, a parameter p is chosen so that on any search move, the probability of making an *off-heuristic* move instead of the the move that would be chosen by the local search heuristic is p. The addition of noise allows the search to jump to new areas of the search space while finding potential local maxima. The amount of noise however is critical, too little and the search is trapped at local peaks. Too much and the search becomes random, missing the peaks. Mcallester et al. (1997) state that the optimal stochastic noise parameter is dependant on the problem characteristics and on the detail of the search heuristic and that it is difficult and time-consuming to obtain an optimal noise for a given problem domain and that it may be influenced by features of a specific problem. They found that the progress of a search can be characterized by measuring the ratio of the mean to the variance of the objective function during the search and that this can allow tuning of the noise parameter more quickly than by the usual trial and error techniques.

Simulated annealing First described by Kirkpatrick et al. (1983), simulated annealing (SA) is a technique based on the statistical mechanics of cooling of metals. During the search a random *off-heuristic* move is chosen with a probability which depends both on the proximity of the objective for a neighbouring location and on a parameter T, the annealing temperature which is slowly decreased as the run progresses. Early in the search, the value of T is such that larger jumps are taken, later the cooling process allows for only small jumps as the search closes in on the optimal location. In its original form the jump probability was determined by $e^{-\frac{f_o - f_x}{kT}}$ where f_0 is the current objective value, f_x the objective value at a nearby point x and T the annealing temperature. The choice of cooling schedule, i.e. how quickly and in what manner T decreases has some influence on the results and is a parameter which requires tuning.

Alternative techniques have been advanced, include quantum annealing (QA) (Apolloni et al., 1989) based on the quantum tunnelling effect. Whereas in SA, the probability of a move out of a local minimum is dependant on only the height of the barrier and so high barriers can cause the search to become trapped, the tunnelling probablity in QA depends both on the height and thickness of the barrier. QA is an attempt to handle search landscapes where shallow local minima are sourrounded by high, thin barriers as the search is capable of escaping from these traps.

Adaptive noise A problem with using random noise in search is that the noise parameter must be tuned using a combination of domain specific knowledge and trial and error. Further it may well depend critically on a specific problem instance. To overcome this limitation, Hoos (2002) developed an adaptive noise mechanism. They use a metric *noise response* based on the distribution of run-times for typical runs of the search. Using this parameter they adapt the level of noise to the rate of improvement. In the early stages they use greedy search, but as the rate of improvement decreases, described as search stagnation, they increase the noise level. If no improvement is obtained the noise is further increased to try to escape local minima. Once improvement occurs the noise level is then quickly decreased again.

In the context of spacecraft operations scheduling using the ASPEN system, an adaptive noise mechanism based on that already described is employed by Fukunaga et al. (2004). They state that the increase and decrease mechanisms are asymmetric. When stagnation occurs there is a delay in applying the noise, effectively while they build up evidence that there is a problem. Once improvements are detected however, the noise is removed at a faster rate than it was applied. They found that in typical scenarios the adaptive mechanism gave results close to those obtained by the static noise technique, though rarely better. In worst case scenarios, where the static noise technique gave poor results, the adaptive mechanism performed significantly better. They argue that in their domain, it is often better to mitigate against worste-case situations than to always obtain best-case results.

Moving search In robotics a common planning problem involves finding a route between various nodes or destinations often with variable costs for the different route decisions. This can be extended to situations where some of the nodes move around or the costs vary over time. The problem is similar in nature to scheduling scenarios in which the goals vary in time. A number of search techniques have been developed for these problems including A^* (Hart et al., 1968), a gradient descent method based on estimated distance to target and cost so far, $LRTA^*$ (Ishida and Shimbo, 1996) an improved real-time version of A^* which learns the costs of visited edges and uses this information to speed up the search. Moving Target Search (MTS) (Yokoo and Ishida, 1999) is an enhanced algorithm in which the *hunter* builds up heuristic knowledge of the edge costs to all the target locations as the *prey* moves between them.

AMP Adaptive Memory Programming (Taillard et al., 1998) was first coined as a general term for a class of optimization techniques which use some form of memory to retain knowledge about poor parts of the search space which have been visited recently.

In TABU search (Glover, 1996), as the space around a solution is searched a short term memory keeps track of locations which have been visited and which have not yielded optimal solutions, typically local minima, to avoid re-tracing into these areas.

In scatter search (Glover et al., 2003) new, fitter solutions are generated by evolving previously found solutions in a process similar to that used in Genetic Algorithms (GA) in that parts of 'fit' solutions are combined to yield new solutions which may be expected to retain some of the *fitness* of the parents. Unlike GAs, scatter search provides mechanisms for combining solutions using derived rules and an adaptive memory rather than making these decisions randomly.

2.2 Case studies

2.2.1 SPIKE

The SPIKE scheduler (Johnston and Miller, 1994) was originally developed for operation of the Hubble Space Telescope (HST). This is a complex scheduling problem involving the allocation of observing time to between 10000 and 30000 observations per year. There are a large number of operational constraints with timescales spanning 6 orders of magnitude.

Planning and scheduling is split into two timescales. The long-term planner assigns activities to a week or part of a week in the HST operating cycle. The short-term scheduler makes the detailed assignment of activities within each week. The scheduler takes account of hard (feasibility) and soft (preference) constraints through the use of *suitability* functions, numerical representations of the degree of preference of one constraint over another.

SPIKE uses an iterative technique described as *Multi-start Stochastic Repair* which operates a cycle of: *Trial assignment* → *Repair* → *Deconflict*. During the *Trial assignment* phase, activities are assigned to time slots using simple heuristics such as *MinConflicts* (Minton et al., 1992). This results in a first schedule attempt which will most likely contain numerous constraint violations and will be sub-optimal.

The *Repair* phase attempts to eliminate constraint violations using a hill-climbing heuristic applied via a neural network. Finally, the *Deconflict* phase tackles any activities which are in conflict by using simple retraction heuristics. The process is repeated over many cycles improving the schedule quality each time. Several quality metrics are used to asses the quality of schedules. These include: Number of observations, total observing time, summed degree of preference.

2.2.2 MicroBOSS

The MicroBOSS scheduler is described in (Sadeh, 1991). The search cycle employs a look-ahead technique to work out a probabilistic demand profile. An *Operation Ordering* heuristic determines the critical jobs associated with the highest demand periods while a *Reservation Ordering* heuristic ranks the costs incurred by these jobs and makes appropriate reservations. A consistency enforcement rule determines new constraints based on the reservations made in the cycle. Where dead-ends occur in the determination of a schedule, backtracking is employed to escape to a known consistent point.

The micro-opportunistics search heuristics allow for constant revision of strategy during construction and repair. MicroBOSS is able to handle reactive repairs to an already partially executed schedule. The operations which need re-scheduling are removed from the existing schedule and new constraints determined based on the existing scheduled activities. The new sub-problem is submitted to microBOSS which determines a solution to add to the existing (running) schedule. In tests conducted against a series of despatching rules on a jobshop problem, MicroBOSS was found to be less expensive (around 20%) in terms of the chosen cost function - a combination of tardiness and inventory costs.

2.2.3 Conflict Partition Scheduling

Muscettola (1992) describes a system (CPS) for solving scheduling problems by identifying regions of the search space where bottleneck conflicts occur, then posting constraints to move the search away from these regions where solutions are unlikely to be found. A bottleneck is defined as a neighbourhood in the search space where the time assignment strategy generates a maximum of inconsistency. These are detected by running a number of stochastic simulations to generate resource allocations with time flexibility. The bottlenecks are identified as those points where the most resource contention occurs. Additional sequencing constraints are then posted to reduce this contention.

They employ two measures of contention. *Activity demand* $\Delta(\tau, t_i)$ measures how much an activity τ relies on a time slot t_i by counting the number of simulations in which τ was asssigned to t_i. *Resource contention* $X(\rho, t_j)$ measures how many activities are competing for a resource ρ at time t_j by counting the number of simulations in which ρ is requested during t_j. During this *Capacity analysis* phase, various variable and value ordering heuristics are employed. These heuristics can be altered to suit the scheduling preferences to be enforced such as a preference towards early finishing.

They test against a micro opportunistic scheduler *Microboss* (Sadeh, 1991) and against *Min Conflicts Iterative Repair* (Minton et al., 1992). Relative to *Microboss*, CPS produced more solutions, faster in the harder problem categories though on simpler problems it was slower to converge. Relative to *MinConflicts*, CPS performed consistently better. In harder problem cases with more bottlenecks *MinConflicts* performed very poorly in comparison to both *Microboss* and CPS.

2.2.4 Gerry

Zweben et al. (1994) describe the GERRY system for scheduling space shuttle ground operations. They define 3 types of constraint:- (*i*) temporal constraints represent precedances between activities, (*ii*) resource constraints represent usage of resources and (*iii*) state constraints represent particular environmental state variable assignments required by some activities - certain activities, denoted as *achievers* are able to set these variables. A weighted penalty function is used to measure the cost of constraint violation. Their repair procedure considers each type of constraint seperately and handles repair of N of each type per cycle before moving onto the next cycle. In order to avoid trapping at local optima in the search space they employ simulated annealing to determine acceptance of a newly generated schedule. At each iteration the cost of the current schedule s is compared to the best so far s^* and is accepted with a probability

$P(s, s^*) = \exp -\frac{|cost(s) - cost(s^*)|}{T}$ where T is the *annealing temperature* which is cooled during the search.

To resolve resource constraints, tasks are selected for repair using 3 heuristic criteria:- (*i*) *fitness* - move the task whose resource requirements match the amount of over-allocation most closely - the logic here is that a task which has a small resource requirement is less likely to have much effect, one which has a very large requirement will cause problems wherever it gets moved to, (*ii*) *dependency* - move the task which has the fewest temporal dependants - a task with a large number of dependencies will likely cause additional violations when it is moved and disrupt existing assignments, (*iii*) *distance* - move the task which needs the smallest move to resolve the conflict - a large move is likely to perturb the overall schedule more.The results of these metrics are scored and a task selected for the move. State constraints are repaired using a selection of 5 methods in priority order which involve moving the affected task and/or adding *achiever* tasks into the schedule before the affected task to set the variable appropriately. The GERRY scheduler was found to be very effective in the chosen domain and was incorporated into the NASA Ground Processing Scheduling System (GPSS) an interactive tool for scheduling repair and refurbishment of the space shuttles between missions.

2.2.5 OPIS/OZONE

Smith (1995) describes the OPortunistic Intelligent Scheduler (OPIS) system. This introduces multi-perspective scheduling in which a number of complimentary schedule repair techniques are employed under the supervision of a Top Level Manager (TLM) and working through a common blackboard representation of the current solution and constraints. External events (changes to requirements, feedback from execution) are fed into the blackboard via model update agents.

Conflict classes are defined relative to a number of conflict metrics including:- conflict duration, conflict size, resource idle time, upstream slack, projected lateness. A number of agents analyse the conflicts which are then matched to fuzzy behavioural profiles. Schedule repair agents are then selected to apply an appropriate repair heuristic suited to the character of the conflicts. e.g. for a problem with HIGH value of *conflict duration* and LOW value of *variance in projected lateness* coupled with HIGH value of *idle time* the *order-scheduling* heuristic is chosen which revises the sequencing of contiguous operations.

In the trade-off between opportunistic improvement and non-disruption to current baseline OPIS is biased towards the latter though this is a function of the analysis and repair heuristics chosen.

Later work on DITOPS (Smith et al., 1996; Smith and Lassila, 1994) an air transport scheduler led to the extension of OPIS into a pluggable object oriented framework OZONE (Object Oriented OPIS = O^3).

For scheduling inflight refueling and transport missions the AMC BarrelMaster scheduler (Smith et al., 2004) was developed using OZONE. In its normal mode of operation the scheduler has to assign times to new missions into an already built schedule. The search strategy `AssignMission` is based on a triple of:-

$Gen_{Resources}$ selects candidate resources (aircraft, crew).

$Gen_{Intervals}$ selects candidate intervals for a mission

$Eval_{Criterion}$ ranks alternatives.

The mission requirements generally lead to heavy over-subscription of resources so various relaxation regimes can be considered:- over-allocation of reserved resources, allowable delays, mission combinations, priority pre-emption (bumping). These are handled by selection of different pluggable combinations of these procedures. e.g. $Eval_{criterion}$ has implementations $Eval_{MinTardiness}$ $Eval_{MinFlyingTime}$ and $Eval_{MinOverAllocation}$, similarly there are several versions of $Gen_{Resources}$ and $Gen_{Intervals}$. A procedure `CombineMissions` allows pairs of missions to be combined to attempt a reduction in resource usage, this can be applied recursively to maximize the reduction in overall flying time required.

The primary goal of the AMC Allocator is to assign the maximum number of high priority missions, often lower priority missions will be left out even though some assigned high priority missions with greater flexibility are included. An incremental optimization procedure `MissionSwap` can be applied to try and insert unassigned low priority missions into the schedule by retracting existing commitments and reassigning to free up slots.

Kramer and Smith (2003) describe 3 heuristics which can be used to select the tasks for retraction:-

- $MaxFlex$ measures the ratio of required time to available time summed over all resources required by a mission and is an indicator of the flexibility of the mission.
- $MinConflicts$ (Minton et al., 1992) measures the number of resource conflicts over a mission's execution interval.
- $MinContention$ which is defined as the ratio of the sum of conflict durations to total required time, measures the proportion of a mission's required interval that is in conflict

The technique is extended (Kramer and Smith, 2004) to minimize disruption to the existing schedule and to speed up the process by search tree pruning.

2.2.6 ASPEN/CASPER

Rabideau et al. (1999) describes work by NASA JPL on the ASPEN scheduling framework. This is a constraint based search/optimization system intended for spacecraft operations scheduling. During the constraint satisfaction cycle activities are slotted into the schedule and conflicts detected. The system then classifies these conflicts into a large set depending on the type of constraint broken or the type of resource bottleneck. A prioritized sequence of repair heuristics is then selected in turn to attempt a repair which moves closer to satisfying.

ASPEN allows the specification of a number of search heuristics to be slotted in at decision points in the algorithm. Some generic ones described include:- (*i*) *conflict sorting* heuristic (a variable order heuristic)- prefers to repair conflicts which require adding new activities, (*ii*) *repair selection* heuristic - prefers to move an activity then adding activities then deleting activities, (*iii*) *interval selection* heuristic for activities being created or moved (a value order heuristic)- prefers intervals which do not create new conflicts then intervals which minimize new conflicts .

Rabideau et al. (2000) move on to describe an extension to ASPEN to allow interleaved repair and optimization using *experts* - these are software components implementing heuristic operations *"an expert is a link between changes in the plan and the change in quality"*.

A number of classes of user preferences are defined, some acting on a local level, others globally. These specify a mapping from local variables to scoring metrics. An example given is of a preference on the start time of one activity relative to the preceding one centred on a *preferred* time gap and decreasing monotonically either side within cutoff limits. Basically this can be interpreted as *"I would like a gap of t^* but will be happy with any gap from t_{low} to t_{high}"* .

Improvement experts include:- (*i*) *local activity variable expert* - considers variables which currently contribute low values to the score. The preferences allow this expert to decide which way the variable has to be adjusted to increase the score (e.g. for the gap preference above, the direction is towards the *preferred* time gap entailing moving one of the activities backward or the other forward), (*ii*) *activity/goal count expert* - aims to increase or decrease the number of activities of a given type. The only tactic for this expert is to add or remove activities, (*iii*) *resource/state variable expert* -

tries to improve the preference scores for resource variables. This can involve moving activities to increase/decrease resource usage (e.g. battery minimum level) or adding and removing activities which increase/decrease the resource level, (*iv*) *resource/state change count expert* - is tasked with improving the score for numbers of state or resource changes, (*v*) *state duration expert* - can move, add or delete activities which maintain or cause a particular preferred state.

In the optimization phase a monotonic increasing assumption is made - i.e. only make (local) changes which will improve globally. A variable order heuristic selects either the lowest scoring preference or the one with the highest potential gain.

In order to improve overall performance an adaptive noise mechanism following Hoos (2002) has been implemented for ASPEN (Fukunaga et al., 2004) and was added to the *repair selection heuristic* above. Its precept is that if the improvement rate stagnates do some random repair activity, then after improvement back off but at a faster rate.

The CASPER system (Chien et al., 1999b, 2000) was designed as a *soft, realtime* version of ASPEN to act as a framework for dynamic replanning incorporating the concepts of continuous planning, execution and replanning and incremental plan extension. They suggest that a plan must be continuously modified in light of changing operating context. It advocates a hierarchic system of planning. At higher levels there should be more reasoned plans over long time scales whilst at lower levels short time scales and more reactive behaviour is the order of the day.

CASPER represents the *world* at a given planning horizon as (current goal set, plan, current execution state, model of predicted states). Updates to any of (goal set, current execution state, horizon (i.e. even just time advancing)) causes a replanning iteration. Changes are posted, the effects of these are propagated, including conflict identification, the plan is repaired and a new working plan results.

1. Initialize Plan, Goal-set, State.

2. Update Goal-set to reflect new goals and remove spurious ones.

3. Update State to current execution state.

4. Compute conflicts.

5. Apply conflict resolution to generate new Plan.

6. Release for execution.

7. Repeat.

According to Chien et al. (1999a), the benefits include:- responsiveness to sudden changes in the environment, predictive modelling errors are reduced due to continuous updating, fault protection is moved from the executive layers working on very short time scales and there is a reduced distinction between planning, scheduling and execution due to *de-layering*. Currently the system can only replan at activity boundaries and is unable to model effects of interrupted activities. Extensions are to include pluggable goal achievement modules (GAMs), these are experts at solving specific types of conflict - e.g. spacecraft attitude conflicts.

2.3 Contingency, flexibility and reactive scheduling

During execution, a number of things can go wrong to upset the carefully worked out schedule. New tasks can arrive, resources can fail (e.g. instruments go offline) and deadlines (goals) may change.

The main approaches to solving these problems can be classified according to Policella et al. (2003) as:-

Robust solutions In this approach the aim is to create solutions with a degree of inbuilt flexibility.

Partially defined schedules Here rather than specify what to do and when, a series of alternative futures or partial sequences are constructed. At execution time the scheduler can switch between these as required.

Reactive rescheduling Rather than trying to predict what might occur or build in flexibility, the reactive approach assumes the schedule will run correctly. When something goes wrong, the scheduler must make some sort of repair to the remaining schedule which might include a complete rebuild of the future sequence.

Dynamic despatch Rather than building an initial schedule, dynamic despatching involves making a single scheduling decision at a time based on the current conditions.

Two classes of reactive approach are described by Jones and Rabelo (1998).

Reactive repair In this type of system the scheduler waits for an event to occur before attempting to recover the system. In the context of a steel-making plant Dorn et al. (1995) discuss a scheduling system in which fuzzy distributions of constraint satisfaction and importance of jobs are used to create a schedule using an iterative improvement technique based on TABU search. The resulting schedule has some degree of flexibility in the ranges of start time and duration for jobs and is thus effectively more robust to changes. In reaction to unexpected events (e.g. a job finishing early) the same improvement technique is used to project the change forward to yield a better schedule.

Pro-active adjustment During execution the system monitors progress continuously, predicting future evolution and attempting to plan ahead for contingencies while the plan is executing. To solve a problem in scheduling batch production in a chemical plant with variable execution times Sanmart et al. (1996) developed the Projected Operation Modification Algorithm (POMA). This involves making estimates ahead in time of the duration of the tasks, then as the earlier tasks are completed, comparing actual times with the previous estimates. The information is used to modify start times of the later, as yet unexecuted tasks and was found to improve overall plant efficiency. They suggest that a more integrated approach to scheduling, schedule modification and plant control is warranted.

Three pro-active techniques for building extra time into a schedule to cope with uncertain duration are compared by Davenport et al. (2001). (*i*) *Temporal protection* adds a slack time into each activity duration prior to the search, (*ii*) *time slack window* uses reasoning during the search to attach minimum slack into each activity, (*iii*) *focused time window slack* assigns slack based on the distance along the planning horizon.

When tested on simulated problems with varying degrees of uncertainty (breakdowns) all methods help improve tardiness with increasing degrees of uncertainty. They do not however give account of the tradeoff due to unnecessary slack time introduced by the technique relative to the gains of continuity and reduced need for rescheduling.

Two different approaches to building schedules with flexibility are compared by Policella et al. (2003) and described in more detail (Policella, 2005). They define 3 measures of robustness - (*i*) reactiveness - speed of response, (*ii*) stability - degree of change induced by reaction, a ripple effect, (*iii*) solution quality - preservation (or enhancement) of performance relative to baseline schedule..

Two metrics are defined for evaluating schedule quality:-

$$fldt = \sum_{i=1}^{n} \sum_{j \neq i}^{n} \frac{|d(e_i, s_j) - d(s_i, e_j)|}{H \times n \times (n-1)} \tag{1}$$

where $d(x, y)$ is the distance between x and y and e_i is the finish time for activity i, s_i is its start time, H is the problem horizon and n the number of activities. This metric is designed to evaluate the fluidity of the schedule i.e. its ability to absorb time variations. Small values of $fldt$ imply that effects will be localized rather than ripple through the schedule. The second metric:-

$$dsrp = \frac{1}{n}\sum_{i=1}^{n} P_{dis}(a_i)\frac{slack_i}{num_{changes}(a_i, \Delta a_i)} \tag{2}$$

where $slack_i$ is the slack available to activity i and $num_{changes}(x, y)$ is the number of activities moved from their start times when activity x is delayed by y with $P_{dis}(a_i)$ estimated as $\frac{duration_i}{makespan}$. This metric describes the disruptibility of the schedule and they claim it measures *"the price to pay for the flexibility of the schedule"*.

The first technique *resource envelope based* employs a 2 step process. In the first step, from an initial partially ordered schedule with constraints they compute the resource-envelope (a time varying measure of resource requirements). Using this they then detect conflicts (where more activities require a resource than its capacity allows). A selection heuristic is used to rank and then select a pair of competing activities. A sequencing heuristic then specifies (posts) new precedence constraints to remove this conflict. The resulting modified schedule with new constraints is fed back into the first step until a solution is found.

The second technique *earliest start time* starts with a pre-selected fixed-time schedule, then selecting activities based on ranked order of start times and using a cheaper resource analysis posts new precedence constraints which can be used to determine the bounds for each activity to produce a flexible schedule. They find that the second approach perfoms best against all quality measures and is fastest.

A different approach *Just in Case Scheduling* (JIC) is taken by Bresina et al. (1994); Drummond et al. (1994) to a problem involving selection of observations for a telescope operated using the ATIS control system in which the main source of uncertainty relates to the lengths of observations (action duration uncertainty). The uncertainty is due to star-centring (acquisition) which depends on sky conditions, wind and pointing accuracy. Uncertainty grows with time. Online scheduling is slow as they have a whole night's observations to allocate to enablement intervals.

They run multiple simulations over the night looking for the most likely break points then look for alternative branches to execute according to the following procedure.

1. Estimate temporal uncertainty at each activity point.
2. Find most probably breakpoint.
3. Create branch (do X or nodo X).
4. Reschedule subproblem (doall before X, nodo X)
5. Integrate with prior schedule.
6. Repeat while time left.

A later related problem in the context of Mars rover operations is described by Bresina et al. (1999). In this case the task is to create a multiply contingent schedule from a fixed schedule in a reasonable time in order to increase robustness. There is a particularly large search space as almost any scheduled action can break so there is a need to reduce the number of branch points to a manageable size. JIC was improved by adding additional resource uncertainty, other than just *time*. An expected utility measure is used to select likely branches and a biased search mechanism with noise, Heuristic Biased Stochastic Sampling (HBSS) (Bresina, 1996) was also introduced.

2.4 Agent based scheduling

Agents are software entities which are capable of acting with a degree of autonomy on behalf of a user or other software entity. They are often goal-driven and may exhibit a number of traits including: learning from past experiences, self-organization and cooperation with other agents. When used in a scheduling context, agent based systems have typically involved multiple agents cooperating to solve the problem. Much of the research describes auction and negotiation mechanisms used to achieve a global solution.

A human-agent based cooperative scheduling system is described by Murthy et al. (1997). The intelligent agents, which are experts in various different aspects of the problem domain construct feasible schedules through cooperative efforts. The agents are classified into 3 seperate categories: *constructors* create the initial solutions, *improvers* modify these acccording to their goals, *destroyers* selectively remove bad solutions. The final selection is made by a human scheduler who may further adapt the candidate solutions.

Scheduling of transportation orders by a fleet of trucks is considered by Mes et al. (2007). The problem domain is a volatile one in which the goals change frequently as new orders arrive, delivery requirements alter and vehicles are delayed. Their solution involves assigning agents to each job and to each vehicle. The job agents request bids from the vehicle agents via a Vickrey auction (second price, sealed bid). In order to calculate bids, the vehicle agents take into account the additional time the change to their schedule will take along with expected waiting time and additonal penalties for lateness. The vehicle agents use a repertoire of techniques to re-arrange their schedules in order to bid but as the individual vehicle schedules are short this is not too costly. The job agents use 2 mechanisms for selecting the winning bid.

In the first technique they simply pick the winner of the first auction. In the second method they are able to reject bids if the prices seem high (they obtain this information from analysis of previous auctions). They may then, if there is sufficient time before the due time of the job, run a second or further auctions to try for a better price. They compare their work against two more traditional central planning systems in which planners re-schedule all the vehicles in response to changing circumstances. They find overall an improvement in vehicle usage and more stable service levels. A further enhancement to performance was found when vehicle agents were allowed to exchange jobs.

A distributed scheduling system based on negotiation between agents is presented by Chun and Wong (2003). They use a modified version of the A^* search algorithm (N^*) which employs a variety of negotiation strategies to arrive at solutions to meeting scheduling problems by maximizing the average preference levels of the agents. In this system they expect that agents will not be willing to disclose full details of their preferences to other agents so no central authority can be expected to arrive at an optimal solution in terms of the various agent preferences.

A subset of the agent based scheduling research has concentrated on the idea of using market forces to determine schedules. In such cases the agents are acting as economic agents bidding for time or resources. In a manufacturing system where agents bid for a series of time slots with constraints on the number and time of the last slot to achieve a particular product manufacturing task, MacKie-Mason and Wellman (2004) modified the bidding procedure of their agents by including a price-prediction strategy. Using this strategy the agents are able to make more informed decisions on when and how much to bid for time slots. In this problem an agent can loose both time and money by bidding for slots in the hope of getting sufficient numbers of these to complete the task. The strategy is implemented by agents observing prices in previous auctions then making predictions on how these will develop. As the auctions run, the agents observe whether their predictions are accurate, even if they have or had no intention of bidding for these particular slots. Their price predictions are then modified in light of this information. The agents bidding policy, both which auctions to take part in and how much to bid in them are determined by the expected payoffs computed from the information it has learnt.

Allan et al. (2006) describe a system (eSTAR) for using intelligent agents to request observations on telescopes. An adaptive scheduling agent system (ASA) has been employed on the HTN network (Naylor et al., 2006), a set of 3 telescopes including the LT by Saunders et al. (2006, 2008) to schedule observations of undersampled, periodic time-varying sources.

2.5 Evolutionary and biologically inspired techniques

Over the last few decades (at least since mid 1950s) a number of computational techniques based on biological analogs have emerged. These *biomimetic* techniques including Genetic Algorithms (GA), Artificial Neural networks (ANN), Artificial Immune Systems (AIS), Ant Colony Optimization (ACO) and Particle Swarms seek to model the behaviour of their biological analogs not in order to better understand the biological systems but to produce solutions to complex problems. Biological systems in general are based on a small set of building blocks and the application of an equally small set of simple rules, yet from these systems emerge highly complex and adaptive behaviour.

Genetic Algorithms (GA), first described by Fraser (1957) are a method of solving optimization problems by mimicing genetic evolution. Candidate solutions are encoded as strings of components (genes). After creating an initial population of solutions, a series of evolution steps is followed in which candidates are *mated* to produce offspring, some of which are likely to be better in terms of some objective function. Mating is performed by splitting each of the parent strings at some point then joining the left and right sections to one another in analogy to genetic crossover. At each evolutionary step many of the poorer solutions are culled in analogy of the concept *survival of the fittest*. Occasionally mutations to some of the solutions are performed where one or more components are changed at random. In effect the technique is a search mechanism. The crossover component is a form of local hill-climbing while the mutation provides a jump to a new area of the search space to avoid trapping in local minima. This is summed up by Russell and Norvig (2003) who describes GAs as a form of *"stochastic hill-climbing search with random exploration and exchange of information between parallel search threads"*.

Perhaps the most unusual use of GAs in scheduling is described by Hart et al. (1999). The problem involves scheduling groups of chicken catching squads and transportation vehicles to deliver a steady stream of chickens to 2 processing factories. They split the problem into 2 subtasks each solved using GAs. The first problem was the assignment of tasks to squads. Once this was achieved a second GA was used to generate the start times of the tasks. Overall they found that the GA scheduler achieved similar results in terms of quality to the existing human scheduler though in much faster times and the overall requirements for vehicles was reduced.

Genetic algorithms are used by Pernin et al. (2008) to solve battlefield manouevering problems (avenues of approach). A two stage technique was proposed in which a GA first solves the route finding problem over complex terrain by stochastically searching a vaste space of possible paths. A second GA is then used to allocate forces.

Artificial Immune Systems (AIS) were first studied by Ishida (1990). These are attempts to model the natural vertebrate immune system to yield novel solutions to computing problems. The immune system is naturally adaptive and posseses further desirable properties such as: learning, feature extraction, memory, self-organization and is highly distributed.

In biological systems, foreign objects (antigens) are recognized by the body and stimulate an immune response. Specialized *B-cells* produce antibodies by a process of cloning and mutation to match and bind to the antigens. From a relatively small initial set of components, a huge range of antibodies can be evolved quickly to deal with new and unforeseen attacks. The theory of the *Immune Network*, first proposed by Jerne (1974) (who later recieved the 1984 Nobel Prize (Jerne, 1985) for this work) suggests that B-cells co-stimulate each other mimicing the effect of antigen receptors. In this way a memory is built up in which B-cells which have been useful in combating specific attacks are reinforced while those which have been of less use are suppressed, leading to self-regulation.

A summary of applications of AIS to date was made by Timmis et al. (2004) and includes many potential areas of applicability including its use in the field of scheduling.

The use of AIS in solving scheduling problems in a factory context was pioneered by Mori et al. (1994). Later work by Hart et al. (1998) and Hart and Ross (1999) on the use of immune system analogues to model scheduling in a rapidly changing environment lead to a system capable of producing adaptive schedules. In their system, schedules are modelled as antibodies while changes to the environment (schedule objectives) are modelled as antigens. The two phase process consists of firstly using a genetic algorithm to generate a large number of possible schedules (the antibody pool) to cover as many possible environment scenarios as possible. In the second phase, changing patterns in the environment (antigens) leading to the requirement of new or modified schedules were classified and used to select an appropriate (antibody) response, namely the new or modifed schedule.

A summary of work on the application of AIS to scheduling to date is presented by Darmoul et al. (2006). It concludes the subject is still very much in its infancy and that there is significant potential for further advances.

Features of the behaviour of social insects have been studied by several researchers in an attempt to solve various optimization and planning problems. A good summary of the range of these techniques applied in the sphere of multi-agent systems is found in Parunak (1997). He describes these natural systems in terms of the desired system behaviour that emerges from the actions of the community with no central controller. Diverse species are discussed including:

- ant colonies from the point of view of foraging as a technique to apply to minimum graph-spanning problems and brood sorting as a distributed sorting method.
- termite nest building as an example of complex construction by reinforcement of behaviour by pheromone signals.
- wasps as an example of task differentiation and specialization by a combination of reinforcement learning and tournament competition.
- bird/fish flocking as an example of local coordination - the birds respond to the movements of nearest neighbours and exhibit an overall coordinated flight though no communication or control takes place.

Ant colony optimization (ACO) algorithms, originally described by Dorigo (1992) are based on the foraging behaviour of species of ants. Worker ants out foraging randomly for food leave behind them pheromone trails back to the colony once they have found food. The trails deteriorate with time by evaporation. When other ants discover a pheromone trail they are likely to follow it. The likelihood of following being increased the stronger the trail. As more ants follow a promising trail they leave behind additional pheromone thus strengthening it. Ultimately the most promising solutions are found along the strongest trails. This indirect communication process, where the agents modify the environment so that other agents perceive these changes is described as stigmergy (Grass, 1959). The technique permits a multiple parallel search of a complex space with reinforcement (positive feedback of the trails) and through the evaporation of the pheromones, the ability to jump out of local minima. Dorigo and Gambardella (1995) employ this technique in solving the travelling salesman problem (TSP), a classic routing problem.

The ants solve the problem by building partial solutions in a constructive manner guided by local search heuristics and cooperating by the laying of pheromone trails as good partial solutions are found. They find that the method compares favourably with other evolutionary methods but is less effective for this problem than specific local search methods. The ACO paradigm has been extended by Bell and McMullen (2004) to solve a multiple vehicle routing problem (VRP), a variant of the TSP.

They modify the standard ACO algorithm by adding an exchange mechanism prior to the trail updating phase. In this they allow permutations of route segments to be tested for fitness. A second improvement involves producing candidate lists for each step rather than taking random movements. Overall they find their ACO is able to solve the VRP to within 1% of known optimal solutions for smaller problems but the improvement does not scale for larger problems. They suggest that using multiple colonies would be the most promising direction for the research to go in larger problems. Gambardella et al. (1999) used a multi-colony approach to solving the VRP with additional time-window constraints. They employ seperate colonies with different objectives. One colony aims to construct shortest tours. The second colony aims to maximize the number of customers visted. The two colonies lay their own pheromone trails but are able to exchange information.

Particle swarm optimization (Kennedy and Eberhart, 1995) is a population based stochastic optimization technique for the solution of continuous optimization problems. It is inspired by social behaviors in flocks of birds and schools of fish. In particle swarm optimization (PSO), a set of software agents called particles search for good solutions to a given continuous optimization problem. Each particle is a solution of the considered problem and uses its own experience and the experience of neighbor particles to choose how to move in the search space.

Starting at a random initial position in the search space and with a random initial velocity, the particles are allowed to move about the search space memorizing the best positions they have found so far. On each iteration the particle modifies its velocity using 3 components:- (*i*) its current velocity, (*ii*) a component driving it towards it best location found so far, (*iii*) a component driving it towards the best location found by neighbouring particles.

This operation of a particle swarm optimizer is summed up succinctly by Kennedy (1997).

> *A common belief amongst researchers is that the swarm behaviour varies between exploratory behaviour, that is, searching a broader region of the search-space, and exploitative behaviour, that is, a locally oriented search so as to get closer to a (possibly local) optimum. This school of thought has been prevalent since the inception of PSO. This school of thought contends that the PSO algorithm and its parameters must be chosen so as to properly balance between exploration and exploitation to avoid premature convergence to a local optimum yet still ensure a good rate of convergence to the optimum.*

2.6 Adaptive Learning techniques applied to scheduling

Reinforcement learning (RL) techniques involve learning policies for state-space problem solving. For each state s, the policy π determines the action a to perform. Whilst learning, the system receives a reinforcement signal or reward after each action. The goal is to find an optimal policy π^* which maximizes the expected cumulative reward over future actions. In scheduling, the policy tells us how (what scheduling action to perform) to maximize some measure of schedule quality in the final realized schedule.

Motivated by the myopism of local despatching rules which lead to supboptimal global behaviour, Riedmiller and Riedmiller (1999) have studied the use of RL techniques to learn despatching rules which adapt dynamically using feedback from the evolving problem situation. The problem is represented as a Markov Decision Process (MDP) where $s(t)$ represents the allocation of tasks to resources at time t and $a(t)$ represents the selection of the next job to allocate. Individual Q-learning agents with local state and action knowledge are associated with each resource. They used a Multi-Layer Preceptron (MLP), a simple type of neural network, to represent the value function, taking as inputs a number of problem features culled from the problem space (set of unallocated tasks). These include features relating to the current schedule state:- tightness with respect to due dates, estimated tardiness, estimated makespan, average slack, and features dependant on the next job selection such as:- average remaining slack if job_i is selected, relative slack ($job_i/total$).

By varying the set of input features selected to match those of standard despatch heuristics (Earliest Due Date (EDD), Shortest Processing Time (SPT), Longest Processing Time (LPT), FIFO and MinSlack (Smith and Cheng, 1993)) they were able to train the network to produce dispatch policies which met or exceeded the performance of these heuristics on problems to which the specific heuristics were best suited. By combining sets of input features they were also able to outperform all of these despatch heuristics on a variety of problems. In effect the network was able to learn better policies by combining the standard heuristics depending on the problem features. When applied to untrained problems the network was able to successfully generalize and improved significantly on each of the standard despatch policies.

Though possible to engineer domain-specific heuristics by hand to exploit regularities and features of a problem space, this can be time consuming and expensive and is naturally non-general. This was the motivation for Zhang and Dietterich (1995) who have studied the use of RL techniques to learn heuristics. Using a temporal difference based technique ($TD(\lambda)$) in which the value function is represented by the weights in a feed-forward network, the reward at each learning step t is computed as the summed relative utilization index for each resource at that time step.

Their system models an iterative repair technique. The states represent the constructed schedule at a point in a sequence of repairs to obtain an optimal schedule and the actions are selected from a set of repair operations. They use a number of features extracted from the partial schedule at each learning step as inputs to the neural network and this is used to estimate the value function. In tests against an iterative repair technique employing stochastic search via SA (Zweben et al., 1994) the system was able to learn a repair policy after training which beat the iterative repair technique consistently for speed though the repair technique was able to produce better schedules given sufficient time.

In dynamic systems, Shaw et al. (1990) hypothesise that the rules used to make scheduling decisions should change with time as the problem characteristics evolve. They proposed a system which distinguishes between and ranks problem characteristics by relative importance, then performs adaptive scheduling by opportunistically selecting appropriate heuristics.

The system called Pattern Directed Scheduling (PDS) works in 2 stages. In the first step (learning stage) a series of training scenarios are simulated and used to study the effects of applying various despatching rules. A critic module (the expert) analyses the performance of these rules on the problem scenarios and may generate new training examples to refine the matching of patterns to rules. The system chosen for induction was based on Iterative Dichotomizer 3 (ID3) from work by Quinlan (1986). In this system a tree of rules is built up by splitting the domains of the problem attributes.

An effect of this system is that it ranks the attributes in terms of an entropy - *"how much does attribute j contribute to the knowledge used to make a given decision?"*. This has the advantage of allowing us to see which attributes are important and which are irrelevant or decision-neutral but does have the disadvantage of considering each attribute in isolation and is unable to detect interdependancies between attributes. A typical example being where a decision should be made based on the similarity of 2 attributes rather than their individual values. Shaw et al used a total of 9 problem attributes and found that 2 of these were irrelevant.

They found that when applying the learnt rules to real problems it was important to reduce the *nervousness* of the system. As the characteristics changed it was neccessary introduce a smoothing component to avoid switching rules too quickly by waiting until the selection count of a new rule had reached a threshold value. They tested the system against a number of standard despatching rules with a collection of problem instances and concluded that there was an overall improvement of around 11.5% in mean tardiness compared to the best of the single rules applied to any of the problem sets. The improvement was attributed to the adaptive selection of rules and the ability to use feedback to refine the heuristic selection.

Case based reasoning (CBR) is a learning technique in which rules are induced by matching problem situations against a set of examples (the cases). It has several advantages:- CBR is particularly useful at extracting rules from noisy data, it operates incrementally building up its knowledge base while working (there is no large expenditure of effort at the start of the process or any need to check consistency between rules as in a rule-based learning), contextual information may be retained in the cases to help human assessors to understand the induced rules.

Due to their interactions and conflicts, it is often difficult to determine numerically or in terms of hard-and-fast rules, the relative ranking and trade-offs between users' schedule optimization preferences. The CABINS system (Miyashita and Sycara, 1995) uses CBR to capture these preferences. CABINS provides a framework for acquiring preferences then uses the case base to improve schedules and provide a reactive repair mechanism in response to unforseen events.

The system operates in 2 stages:-

1. In the first stage, a feasible but sub-optimal schedule is generated using a constructive technique.

2. In the second stage the schedule is improved by selecting repair actions (iterative repair). The quality of the schedule before and after each repair are compared using a number of local (pertaining to the current *focal* activity) and global (referring to the overall schedule) criteria (e.g. tardiness, work-in-progress inventory, waiting time). Repair operations are interleaved with consistency enforcement. As a repair on the currently selected *focal* activity is made it is likely that constraints may be broken requiring other activities (conflict set) to be rescheduled.

   CABINS has 3 operating modes:-

   - In *knowledge acquisition* mode the user selects the repair actions to perform and these decisions are stored along with information to characterize the current problem situation (a case). If sufficient training examples are provided, the resulting case base should contain a distribution of examples covering a diverse set of problem situations.
   - In *decision support* mode, the system selects repair actions by matching the current problem to the repair actions in the case base and an interactive user has the option to veto/override providing additional training.
   - In *automatic* mode, the system makes all repair decisions using the case base without user interaction.

Selection of repair actions is performed by matching the problem profile against the stored cases using a k-nearest neighbour matching algorithm:-

$$d_i = \sum_j (s_j^i (\frac{C_j^i - P_j}{E_{dev_j}}))^2 \tag{3}$$

$$z_i = \exp^{-d_i} \tag{4}$$

where s_j^i represents the user's evaluation of the importance (saliance) of case feature j of case i, C_j^i is the value of feature j of case i, P_j is the value of feature j in the current problem and E_{dev_j} is the standard deviation of feature j for all cases. d_i is the dissimilarity between the current problem and the i^{th} case and z_i is the similarity between the current problem and the i^{th} case.

The repair process operates as follows:-

1. A focal activity is selected and a start time predicted using each of the available tactics.

2. The conflict set is worked out by projecting the (ripple) effects of the repair onto neighbouring activities.

3. Consistency enforcement technique works on the conflict set using the *Activity Resource Reliance* (ARR) variable ordering heuristic which selects the most critical activity (most likely to be involved in a capacity conflict over the repair horizon) and a greedy value ordering heuristic which selects a time assignment for the selected activity according to a bias function which represents the time-varing utility perceived for the activity start time deduced from the case base.

4. The activity utility functions are updated - they are biased to start times calculated as part of step (3) to be used in the next iteration.

5. CBR is used to evaluate the quality of the new schedule.

They performed a series of comparisons against other methods evaluated against the following criteria: (*i*) attendance to scheduling objectives, (*ii*) amount of disruption, (*iii*) efficiency (speed) - especially in respect of its use for reactive repair .

Compared with a simulated annealing based iterative repair scheduler, they found that around 1000 cases was optimum. A marginal improvement could be obtained above 1000 but was not worth the effort. They concluded that CABINS was good at capturing preferences and optimization trade-offs that are difficult to model, improved schedule quality irrespectively of how the initial (seed) schedule was generated and produced high quality schedules faster than simliar IR technique so was suitable for reactive scheduling.

Later, Sycara et al. (1995) extended the CABINS framework to consider time-varying user preferences. Their extension allowed the system to learn new cases from its own evaluations of schedule improvement while running. They employed a *rolling horizon* model in which the matching algorithm gives more weight to recent cases than to older cases.

2.7 Summary

As we have seen the scheduling and planning literature covers a wide range of domains (from chicken catching (Hart et al., 1999) to Mars rover operations (Bresina et al., 1999)) and techniques from operations research (Bitner and Reingold, 1975) through to artificial intelligence (Mori et al., 1994). Many of the techniques are domain-specific (Dorn et al., 1995) and not neccessarily readily adapted to other domains. Many of the techniques require tuning to problem domains or even for specific problems (Kirkpatrick et al., 1983). A large body of work is aimed at solving standard/benchmark problems often in the factory production domain. These may need serious adaptation or tuning to be usable in a specific domain or problem instance. Another common theme in many papers involves the production of new metrics which are often specific to the problem domain. However many of the techniques are potentially adaptable to alternative domains or at least provoke thought on how to design new techniques and metrics.

3 Investigation of Metrics

3.1 Introduction and rationale

Metrics are quantitive standards for measuring performance, quality or some aspect of a service or process we wish to monitor. In this section I discuss two distinct classes of metric which will be used in the thesis.

Problem complexity metrics (PCM) measure the difficulty of a scheduling problem and provide an independant variable against which to assess the performance of schedulers. It might be found for example that a particular scheduler which performs well on a *easy* problems is out-performed by an alternative scheduler when faced with *difficult* problems.

Schedule quality metrics (SQM) measure the actual performance of schedulers on such problems. In other words they tell us how well or efficiently a particular scheduler performs.

A third set of metrics which will not be discussed in this section are those metrics actually used by the schedulers to perform scoring, selection, retraction and other heuristic procedures in the determination of schedules. These will be discussed in the relevant scheduler descriptions (Sect. 5.7.1).

3.2 Problem complexity metrics

These metrics are used to describe the complexity of the scheduling problem at any given time. In addition to their use for assessing the performance of schedulers under test, it is likely that an advanced scheduler could itself make use of these metrics by looking ahead to see where hotspots (difficult scheduling periods) are likely to occur and thus take these into account in its decision making. The basic job of the scheduler is to select and sequence sets of observations from the ODB, consequently, this is where the focus of the investigation begins. The content of the observing database or pool (ODB) at any time defines the set of observations that are available for scheduling at that time. This pool evolves due to the arrival of new observing requests, modification of existing request (e.g. in light of observing results) and removal of spent requests. These modifications which may be made by observers themselves via a Phase 2 tool or automatically by external user agents, occur in principle continuously. The complexity manifests itself through changes in the amount and severity of competition between observing requests for particular times and, over the course of a night, in the overall loading. The following metrics are proposed.

3.2.1 Contention profile C_c

This is the time evolving profile of the number of observation groups which could potentially be scheduled according to their explicit timing constraints. Additional refinements include convolving with the probability of the time actually being available based on likely weather and technical downtime forecasts. The average contention over the course of a night gives an estimate of how overloaded the schedule could potentially be. This static contention profile is a crude measure as it does not take into account the fact that some of the groups which figure in the contention profile later in the night may have already been selected by then and thus need not be considered.

It is found that in general (and explicity Fig. 3.1) if we plot both the predicted contention C_C and the actual contention C_A against time on a given night we typically find that the actual contention follows the prediction at the start of the night. As the night evolves, C_A decreases in step fashion as groups are executed and taken out of the pool. At intervals C_A rises up again as different subsets of groups become available though never generally attaining its predicted level.

A possible mechanism to take this into account would be to introduce a weighted probability of execution on those groups figuring later in the night to reduce the the effect of any early executions. A simple exponential decay might be used with $P_{avail} =$

1 at the start of group's execution window and decay factor based on the group's likely execution. How this might be worked out is alas unclear. More importantly this would likely depend on the contention statistics we are actually trying to work out. Effects of disruptions early in the night will also change the contention later as groups which *ought* to have been executed may still be in contention later.

The dynamic contention profile C_{dc} is perhaps a more realistic measure and is calculated by performing a forward simulation through the night and extracting actual contention in the process. This is of course somewhat dependant on the scheduler used but can be a valuable tool for assessing the degree of competition between groups.

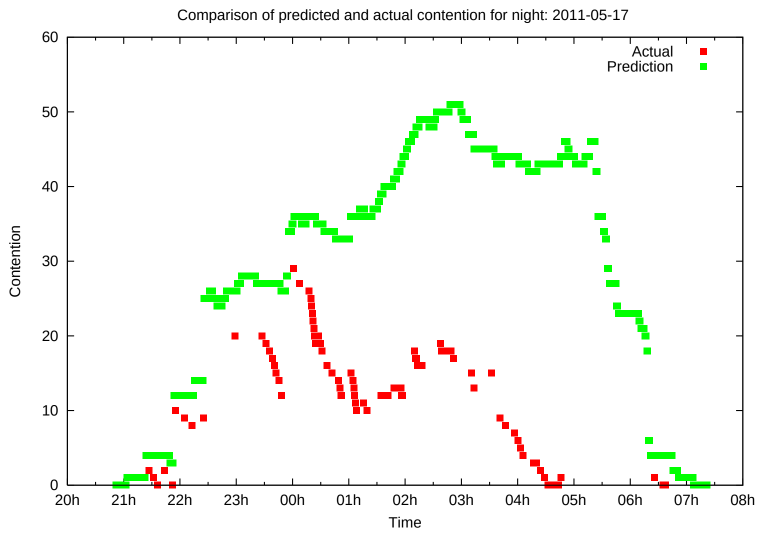


Figure 3.1: Comparison of predicted (C_C) and actual contention (C_A) on a typical night. C_A follows C_C at the start of night and tracks any upward trends asnew groups become observable but over the course of the night C_A slowly drops off. As runs of observations occur, C_A is seen to follow a descending staircase pattern as the pool of observations is depleted. It should also be noted that C_C assumes *ideal* conditions whilst on the night these may vary.

3.2.2 Demand profile C_d

A given group will potentially have multiple observing windows when it should be attempted. During any given window the group can be considered to have a demand C_D on the time within that window. E.g. if the group g has an (estimated) execution time $X(g)$ and its window of opportunity is $\phi(g)$ [1] then the group's demand over that window is $C_D = X/(X + \phi)$. If we add up the demand contributions of all the groups which are enabled at any given time we should have a measure of how much demand is placed on that instant. If this aggregate demand exceeds unity then it is likely that some of the groups will not be observed i.e. the requirement for time is greater than the time available.

There are several refinements that can be made on this estimate. The numerator can be worked out fairly easily. It can usually be considered constant and known. The denominator is more of a problem. Firstly working out the window of opportunity from the group's time constraint window is straightforward, however this window may extend from just a few minutes upto several days or even weeks. In the latter case the group's target(s) may rise and set several times and the various implicit timing constraints may be broken on several occasions. The lunar distance constraint will impose a varying overlap with the target visibility windows. Any solar elevation constraint will vary the length of available night depending on the time of year. If we consider each of the calculable constraints then we can work out the actual amount of time (within the window) that the group can actually be observed. This gives us a revised (increased) estimate for the group's individual demand for those times within the new sub-windows.

Going on another stage we might consider those constraints which cannot be worked out in advance. A group may have a minimum seeing constraint,[2] however, we cannot tell what the seeing will be like at any future time though we may be able to estimate the likelihood of attaining the group's minimum level. Likewise we can obtain estimates of extinction (perhaps including seasonal variation). We should in addition consider the probability that the selected time is even available for observing. Weather and technical downtime mean that a certain fraction of time is lost. A crude climatological estimate might just give the average probability of bad weather (averaged over

[1]this is the window during which it may legally start, it may run on over the end of the window as this is taken account of in determining whether the group could be executed.

[2]minimum seeing in terms of quality and hence largest allowable FWHM.

long periods), we shall later see (Sect. 4.3.2) that this is around 20%. We might also have more accurate seasonal-adjusted climatological information. If we could predict for some time ahead based on current and recently collected weather data then we should have an even better demand estimate.

Formally, let $X(g)$ represent the *estimated* execution time of a group g and $U(g,t)$ the remaining useful time left for that group at time t, then the partial demand of g at time t is defined as:-

$$d(g,t) = \frac{X(g)}{X(g) + U(g,t)} \tag{5}$$

$$U(g,t) = W^{*}(g,t) \cap N(t) \cap M(g) \cap V(g) \tag{6}$$

where $W^{*}(g,t)$ is the set of remaining sub-windows (running forward from t) for the current execution of the group, $V(g)$ is the set of visibility windows for g's targets from t onwards, $M(g)$ is the set of windows satisfying g's observing constraints, $N(t)$ the set of nights from t onwards.

Calculation of demand is a computationally costly exercise. In general it proves convenient to restrict the look-ahead to the current night giving a slightly higher estimate than would otherwise be made.

As an example, for a *typical* flexible group with a duration of say 1 month we need to derive the feasibility at say 5 minute intervals over this period, some 8640 calculations. By preprocessing, e.g. by pre-calculating the target visibility and night periods we can reduce this by upto a factor of around 10 to some 800 calculations. A single feasibility calculation takes around 5ms on the deployed system so for the example group we need around 4s to calculate this metric. For a simulation where we are calculating the metric for upto 500 groups over a night we would require 2000s per simulation run.

3.2.3 Load C_l

Load is a fairly simple metric which describes the ratio of executable time in a given observing night to the length of the night (or astronomical night). Simple load is calculated using the sum of execution times for each executable window of each group that can be executed during the night. It includes all windows of all groups whether they *must* be executed that night or not. An urgency weighted load C_{ul} can also be calculated which weights each execution time by the reciprocal of the number of remaining nights in which the window could be observed. Critical load C_{cl} is a pruned version of the normal load where only groups which **must** be executed that particular night are included. Priority weighted load C_{pl} is calculated by taking into account all of a group's windows which should be executed on a given observing night then weighting by the priority of the group whose window it is.

3.3 Schedule quality metrics

One aspect of the planning and scheduling process is *to maximize the profitability of the enterprise* (Miyashita and Hori, 1996). The enterprise is the telescope or the group which runs it. Profit refers here to some accumulated reward the *owners* expect to get from providing the facility. This could be money, in terms of grants recieved for the continued provision and expansion of the service or could refer to some general sort of kudos within the scientific community resulting from the perception of the current and potential users that it is a novel and useful facility.

The utility or reward can be looked at from 2 (sometimes conflicting) points of view. We need to be able to measure a quantifiable reward resulting from performing the most scientifically useful or *in-vogue* research whilst at the same time meeting the various users' preferences.

.

- *Enterprise utility* $V_e(g,t)$ represents the reward to the telescope or facility. Ideally the enterprise wants to do (scientifically) valuable work and as much of it as possible. Some of the factors which might influence the enterprise view include:-
  - Proposals have already been vetted and prioritized on the basis of scientific importance by the allocation committees so this provides a useful measure.
  - The enterprise does not want to waste high quality time making observations which could be performed in poorer conditions so a measure of matching conditions to minimum requirements.
  - The enterprise may be more inclined to reach particular stages of completion for certain (high valued) projects over other lower valued projects.
  - Conversely there may be a requirement to share out general observing time (and time under various conditions) between projects or sponsoring TAGs in previously negotiated proportions.
  - We may want to factor in any periods of idle time caused by waiting for specific groups to become enabled as a penalty/cost term.
- *User utility* $V_u(g,t)$ represents the reward to the individual users in respect of their own preference metrics. From a user's point of view, the ideal schedule is probably one in which all of that particular user's feasible observations would be done at their optimum times irrespective of any other user's requirements. Of course we cannot always work out these optimal times. We can work out when the target will be best placed, e.g. at its highest elevation in the current window whilst bearing in mind that the window may extend over several nights, but we cannot say what the conditions will be like at that time. The telescope may be out of action due to weather or the seeing could be very poor.

  The individual users will have some preferences on when and under what conditions their observations are performed. They are able to specify the minimum conditions and the timing intervals and various constraints on their observations but these preferences are not ordered in any way - e.g. one user might prefer that their observations are done at low airmass and be less concerned with regularity whilst another user may be less concerned with airmass but more so with regularity. The simple constraints do not allow these preferences to be taken into account, instead a generic preference weighting is applied by the scoring and selection models so that all observations are treated as if the users had the same preference weighting.

The user's preferences metric could include factors such as how close to centre of time window, how good the airmass, how well conditions are/will be matched. Each group could have a set of user-specified weights for these as they would have their own preferences as to which are more important and the relative importance of particular attribute values. Factors which we might include in $V_u(g, t)$ include:-

- airmass
- sky brightness
- seeing matching
- window position
- data profile tracking

With these points in mind, a first attempt at an overall utility measure for a night might look something like Eq. 7.

$$V_{night} = \sum_j ((w_u U_j(t_j) + w_e E_j(t_j)) * x_j) \tag{7}$$

where w_u is the weighting for user's preference satisfaction, U_j is the value the user assigns to the performing of his observation of group j at the selected time t_j, w_e is the weighting for enterprise value, E_j is the value of scientific priority and other enterprise measures assigned to the group j and x_j is the execution time or duration of the j^{th} group.

This metric adds in the contribution from enterprise and user-preference matching but if all users are free to specify their own weightings this might be thought to cause problems with comparison. E.g. If user A assigns preferences so his observations are always good i.e. he has no preference and user B chooses his preferences so his observations are best at t_B, user A will get a better score contribution most of the time. However it should be noted that we are not suggesting use of this measure as a selection metric to select A over B but to merely as a measure of the quality of the schedule for the night.

3.4 Primitive utility measures

Needless to say, choosing the relevant enterprise and user metrics and deciding how these should be weighted is no easy matter and depends on too many factors to come up with a definitive answer which would be of use in the ensuing experiments. Consequently we are forced to find some simpler/more primitive metrics to use. In each of the following metrics the group contribution is weighted by the execution time (X_g) of the group and is scaled by the length of the night.

- The *height* metric Q_H measures the difference between the optimum time for performing a group based on its height to transit height ratio during the night and its actual selected time of execution. By adding these offsets for all selected groups we get a measure of badness. Ideally all groups would be executed at their highest point (in the current feasibility window) so this measure should be close to zero.

- The *optimal height* metric Q_{OH} measures how close to the *optimum* height the group is observed in its execution window.

  The optimum height is basically the highest elevation attained by the group during the feasible observation window - in a given window this may be at the end (rising target), start (setting target) or somewhere in the middle of the observing window (transiting target).

There is however a non-linear relationship between elevation and airmass such that for targets which do not rise particularly high the difference in image quality between worst and best case elevation may be high over a relatively small elevation difference whereas for high rising targets the difference in airmass and hence image quality between best and worst case elevations will be small for the same elevation difference .

Consequently it may be better to measure the ratio of execution-time airmass a_{actual} to best achievable airmass a_{opt} as an Optimal Airmass metric Q_{OA}.

Additionally, the *benefit* (image quality advantage) of one airmass over another may itself be non-linear and this is what we should really be measuring. Q_{BA} measures the ratio of some decreasing benefit function $b(a)$ of airmass a at optimal airmass and actual execution-time airmass. A simple option for $b(a)$ would be to use the known (Sarazin and Roddier, 1990) correction of image FWHM or seeing s for airmass relative to *ideal* seeing at elevation $90°$, $s(a) = s_{90}a^{0.6}$ with s_{90} indicating the image FWHM at $z = 0, a = 1$ and assuming image quality is judged purely (and linearly) on FWHM. It should be noted that this correction depends also on observation wavelength λ.

$$Q_{OA} = \frac{1}{L_N}\sum_{g\in S} X(g,t)\frac{a_{opt}}{a_{actual}} \tag{8}$$

where $a = \sec z$ is the airmass of group's target at zenith distance z.

$$Q_{BA} = \frac{1}{L_N}\sum_{g\in S} X(g,t)\frac{b(a_{actual})}{b(a_{opt})} \tag{9}$$

where $b(a)$ represents the benefit (image quality advantage) for airmass a.

- The *monitor window* metric Q_{MW} measures the difference between the optimum time (relative to the centre of a monitoring window) for performing a group based on the known monitor window and its actual selected execution time. By adding these up we get a measure of badness in terms of offsets from the optimal time. Ideally all groups would be executed at the centre of the monitor window. There are some implications in respect of non-monitor groups. How do we select the centre of the window ? What if the window centre is not actually feasible, how is this taken into account ? In practice these difficulties led to this metric not being used for the experiments but it might be used in future if these problems could be satisfactorily sorted out.

- The *priority* metric Q_{PX} measures the total priority score achieved i.e sums up the priorites of the groups selected. As proposals have already been vetted and prioritized by the allocation committees, this provides a useful source of information.
$$Q_{PX} = \frac{1}{L_N} \sum_{g \in S} X(g,t)p(g) \tag{10}$$
where $p(g)$ is the priority value for group g and $X(g,t)$ is the estimated execution time for group g at execution instant t. In early simulations the priority of a group is a set attribute of the group. In later simulations after a change in the Phase 2 system, it is calculated based on a combination of:- (*i*) priority of the group's proposal assigned by the TAG, (*ii*) timing category and (*iii*) urgency flag .

- The *target demand* metric Q_{TD} measures how *urgent* the selected groups are with reference to the night's demand profile C_d.
$$Q_{TD} = \frac{1}{L_N} \sum_{g \in S} X(g,t)f_D(g,t) \tag{11}$$
where $f_D(g,t)$ represents the value of the demand (Eq. 5) for group g at execution instant t.

- The *execution time* metric Q_{XT} is a measure of the amount of the night during which groups are actively being observed. A low value is effectively an indication of *slack time*.

$$Q_{XT} = \frac{1}{L_N} \sum_{g \in S} X(g,t) \tag{12}$$

- The *remaining nights* metric Q_{RN} counts a decreasing function (for convenience 1/n) of the *number of nights remaining* n for any groups in the observability window $\Phi(g,t)$ at the time of selection/execution. It effectively measures how *urgent* the window was in terms of the number of future chances of executing the group other than tonight. e.g. If a group could be executed tonight or on 3 future nights this yields a value for RN of 4 and thus 1/4 as its urgency.

  $$Q_{RN} = \frac{1}{L_N} \sum_{g \in S} \frac{X(g,t)}{R(g,t,\Phi(g,t))} \tag{13}$$

  where $R(g,t,\phi)$ counts the number of remaining nights for a feasiblity window $\phi(g,t)$ of group g at execution instant t - i.e. it counts the number of feasible future sub-intervals of Φ with maximum of 1 per night.

- The *project completion* Q_{PC} has been suggested as a metric for comparing scheduling policies or schedules generated by different schedulers. I suggest that this is not always a good metric. A project which relies on gathering data over an extended period and for which the length of data gathering, periodicity of gathering are more important would not be well served by completing the program in the fastest time. For such programs we should be more concerned with producing data at a rate which tracks the expected production rate of data.

  There are some problems associated with this metric.

  – In addition we would like to generate the best quality of data or at least data of a quality which meets or exceeds the programs stated requirements - e.g. we select an observation to be done in seeing which is as good or better than required, at low airmass and any other factors which could contribute to its quality.

  – When considering flexibly timed groups it should be noted that these receive all of their data in a single chunk. It is therefore not feasible to *track* these.

  – Finally, we do not always have enough information available early in a semester. Some users put all of their groups in at the start, some feed them in over the semester so we cannot in general know how complete projects are at any time.

- The *condition matching* metrics Q_{SM} (seeing matching) and Q_{LM} (lunar condition matching) assign a measure to how close the actual conditions are to those requested in a group's observing constraints as being the minimum acceptable. These metrics are designed to prevent groups with *low quality* requirements using up time which might be classified as *high quality*.

- The *yield tracking* metric Q_{YT} measures the yield of a group of observations. Put simply, this is the ratio of the quantity of data taken (observations) at a given time relative to the quantity that would have been taken upto that time given that the group was observed at each occasion when it could technically been observed. Calculation of this metric presents some difficulties. In the case of groups with *minimum interval* timing constraints it is not possible to give an actual figure for the potential number of observations made as the constraint does not specifiy a maximum interval. For practical purposes this is taken to be twice the value of the minimum interval. In the case of groups with *flexible* timing constraints, the potential yield is calculated as a fraction of the time between activation and expiry even though there can only be one occasion when the observations are ever made.

- The *resource usage* metrics Q_{Ax} for various resources x against target values. These would probably be long term metrics, valid over periods of the order of a semester or year. An implicit metric currently in use by the operations team is the TAG assigned Minimum Use Fraction (MUF) .

- The *score utility* metric Q_{SU} is a metric generated using a weighted sum of the set of metrics used by the scheduler to derive the scores it uses to select groups for scheduling.

3.5 Matching metrics to scheduling and planning horizons

As can be seen the various Q metrics already discussed cover very different timescales. It is perhaps appropriate then to see how these might be applied in reality to different scales of planning and scheduling. Table. 3.5 is an attempt to match the various Q metrics to the different planning or scheduling horizons discussed in Sect. 1.2

Planning Level	Period	Metrics
Strategic	1 semester or more	YT (yield track), PC (program completion).
Tactical	1 week - 1 month	PC (program completion), Ax (resource sharing).
Mission	1 night	TD (target demand), RN (remaining nights).
Scheduling	1 horizon	OH (optimal height), P (group priority), LM (lunar match), SM (seeing match).

Table 3.1: Possible assignement of Q metrics to appropriate layers of the planning and scheduling hierarchy. As the horizon increases up the hierarchy, the metrics used are likely to be calculated over longer periods of time with possibly large night to night variation. At the scheduling end of the hierarchy, metrics are calculated on a per-group execution basis.

3.6 Conclusions

I have discussed two main types of metric. Quality (or Q) metrics measure the results of scheduling. A realistic Q metric would include both user and enterprise requirements, however for simplicity and because there is currently no way within the Phase 2 system to include user requirements a set of primitive measures have been defined. Complexity (or C) metrics measure the difficulty or *density* of a scheduling problem and are an attempt to capture the highly complex multi-dimensional structure of the Phase 2 information in a few simple parameters. The measures suggested are relatively easy to compute and give a flavour of the problem.

4 Characterization of operating environment

I have already stated (Sect. 1.1) that the telescope operates in an *uncertain environment*. In order to be able to make planning and scheduling decisions over the range of timescales outlined in Sect. 1.2 it will be neccessary to determine what constitute the components of this environment, to ascertain the range and variation of these components and to examine how such variation might impact on both the ability to schedule and the potential reward available.

Though many of these processes are naturally unpredictable, it would be very useful to be able to at least have some way to determine the likeliness of these conditions occurring. If we could determine that the telescope would not be able to observe tomorrow, then we could ensure that an important observation was made tonight even though the potential reward might be better tomorrow. Conversely if we knew for certain that the telescope would be available tomorrow then we might make a low priority but urgent observation tonight and do the less urgent but higher priority observation tomorrow.

4.1 Environment Components

The operating environment characteristics can usefully be broken down into the following components. Each of these can affect the operations of planning and scheduling in particular ways.

- **Technical faults**. Mechanical, electrical, software and communications systems faults can leave the telescope unable to operate or may severely impair operational effectiveness. In some cases the telescope may be deliberately shutdown to permit preventative maintainance.
- **Bad weather**. Rain, high humidity, strong or gusting wind and calima (Saharan dust) can force the shutdown of the telescope systems to prevent damage to delicate instrument, optical and electrical hardware. The presence of ice prevents

the dome from opening due to likely damage to the rams. Under any of these conditions, no observing is feasible.

- **Poor Sky conditions**. Do not pose problems for infrastructure but can severely disrupt the ability to perform useful observations. Extremly bad seeing, high extinction or high background sky-brightness lead to the requirement for longer exposures to achieve required signal-to-noise ratios (SNR). In addition the pool of feasible observations is reduced as the conditions worsen leading to lower contention (easier scheduling task) but at the same time to reduced potential rewards.

- **Phase 2 population**. The actual content of the Phase 2 ODB affects how scheduling decisions are made. The distribution of the large number of its characteristics determines both the contention at any time and the potential rewards available. External software agents and users can modify the content of the Phase2 ODB at any time. Where these modifications occur during the night it seems reasonable to assume they will have some effect on the character and potential profitability of schedules which might be generated and introduce extra complexity into the scheduling operation.

4.2 Technical faults

The telescope is a complex system made up from numerous components. Though it has many inbuilt recovery mechanisms there are times when the overall system is unable to function without direct human intervention. Problems may occur with any of the following systems.

Mechanical Physical problems with components such as:- main axes, focus drive, enclosure hydraulics, science fold deployment and instrument components such as filter-wheels and slides.

Electrical Power supplies to the site are occasionally disrupted though sometimes this information is available in advance.

Network Problems due to internal (inter-system) communications breakdown. These may be as simple as a wire dropping out, a software configuration problem or due to heavy network loading - typically a symptom of some other underlying problem.

Software Low and high level systems can cause problems. These are highly complex systems and thus difficult to test thoroughly even with use of a simulated environment so problems can occur, particularly during upgrades.

Several (7) years of knowledge gained from operating the telescope allow these errors to be classified into a few basic categories with differing timescales from inception to solution.

major catastrophe These are problems which suddenly manifest themselves one night causing major downtime. Often these are due to mechanical breakdown of a component which requires a site visit to repair. The time involved can potentially be days if spares are not immediately available and no workround can be arranged. Where serious software failures occur, these are generally fixed within a day by a concentrated effort.

occasional glitch These are problems due to either non-terminal failure of a mechanical component which is too expensive to justify replacing or more often a known but difficult to repair software problem. The frequency of occurance and degree of impairment will tend to determine how much effort is employed on the solution.

sporadic bug Sometimes the problem is very difficult to identify and needs to occur many times before enough information is gathered to identify the cause.

shakedown Effectively time required for new mechanical components or software deployments to bed in. Software in particular may be tested on the simulator but behave differently on the actual system due to subtle timing effects or unpredicted coincident situations.

4.2.1 Source of data

The main source for this information is the nightly logs kept by telescope operations staff[4]. These consist of the number of hours of technical downtime, weather downtime and actual observing per night. This information is assessed by a combination of examining system logs and from personal knowledge of the events of the previous night and so a degree of interpretation is involved. Of particular importance is the policy to assign the category *weather downtime* to those periods when there was a combination of bad weather and technical problems - it is thus not possible to seperate these out. E.g. on a night where there might be 2 hours of observing mixed with 2 hours of intermittent technical problems followed by 6 hours of bad weather there is no way of telling whether the actual technical problem would have resulted in an additional period of downtime if the weather had remained good.

Operations staff may assess early in the night that some serious technical problem is likely to affect the performance of the telescope during the night - e.g. poor image quality or tracking and decide to shutdown the telescope for the remainder of the night thus yielding an apparently high loss fraction rather than allow the automated systems to cycle between *operational* and *impaired* states.

Data covering a period of 30 months (919 days) for 2005, 2006 and part of 2007 was available and used to determine figures for probability of technical downtime and how this varies over the year.

[4]Obtainable from: http://telescope.livjm.ac.uk/News/Reports/

4.2.2 Analysis

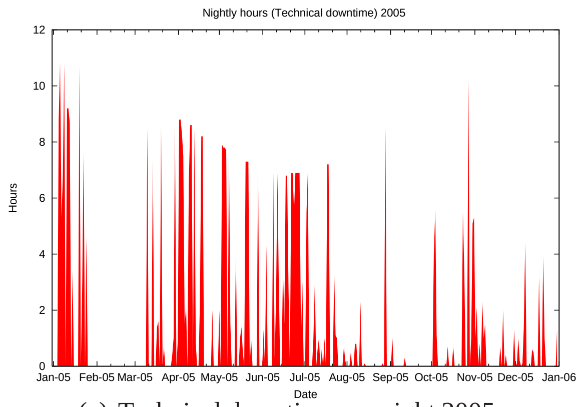
Plots of technical downtime per night are displayed in Figs. 1(a), 1(b) and 1(c). The individual spikes represent the actual number of hours downtime on each specific night while the curved envelope reflects the variable night length over the course of the year. There does not appear to be any discernable pattern though as there is no sub-categorization of actual types of fault one cannot rule out the possibility that certain faults are more common in summer and others in winter, averaging out to a similar overall result. Basically, technical faults of some type can occur at any time.

Fig. 4.2 shows the distribution of fraction of night lost to technical problems, as can bee seen this is bimodal with around 72% of nights suffering less than 5% downtime (including those nights with zero loss). A total of 8% of affected nights over the entire period suffered nearly 100% downtime. The variation of total technical downtime hours averaged by month for the available data is shown in Fig. 4.3. The average monthly downtime is 36.6 hrs over the period corresponding to about 1.2 hours per night.

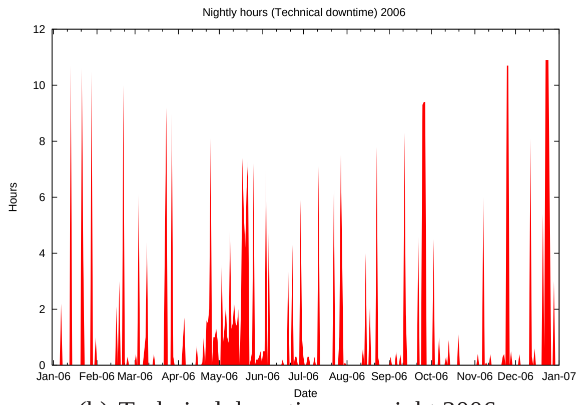
4.2.3 Conclusions

It had been hoped to look at lengths of runs of technical downtime to determine if there were any patterns - e.g. whether these occur in runs and with what distribution and frequency. Contamination of the data from weather downtime stats and the relatively small amount of data available makes this problematic. Heuristically however we might expect this to be the case. A problem might arise sporadically one night causing a degree of lost time, then maybe occurs the following night, spawning attempts by support staff to solve by software or engineering means. This may occur quickly or take several nights to correct, thereafter the problem disappears or becomes less frequent.

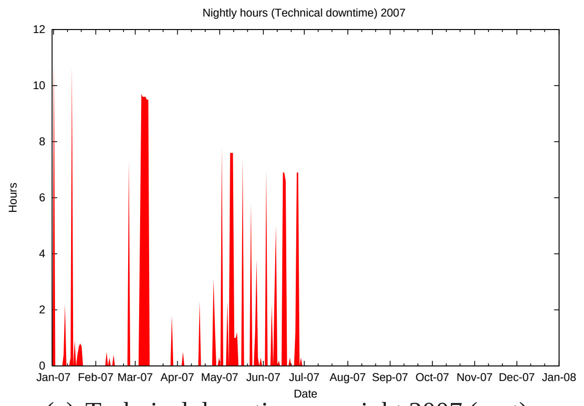
The diverse nature of these problems suggests that there is little chance of predicting future occurances - by their very nature they are unpredictable. The best that can



(a) Technical downtime per night 2005.



(b) Technical downtime per night 2006.



(c) Technical downtime per night 2007 (part).

Figure 4.1: Nightly variation (hours) of technical downtime for years 2005, 2006, 2007(part). Large blocks are empty due to operations policy preference for reporting bad weather downtime rather than technical when both occur simulataneously. The large empty block around February to March 2005 is due to a prolonged period of bad weather where no observing was possible.

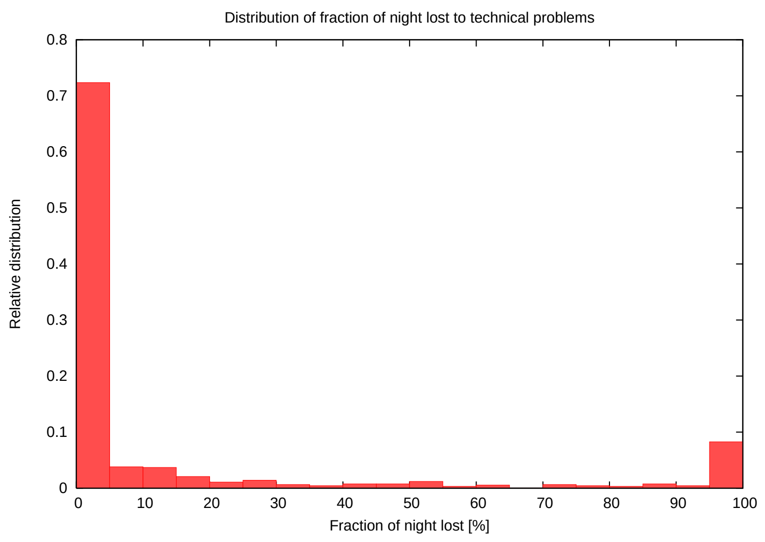


Figure 4.2: Distribution of fraction of night lost to technical problems. Majority of nights (73%) have less than 5% loss, while some 8% suffer near 100% downtime.

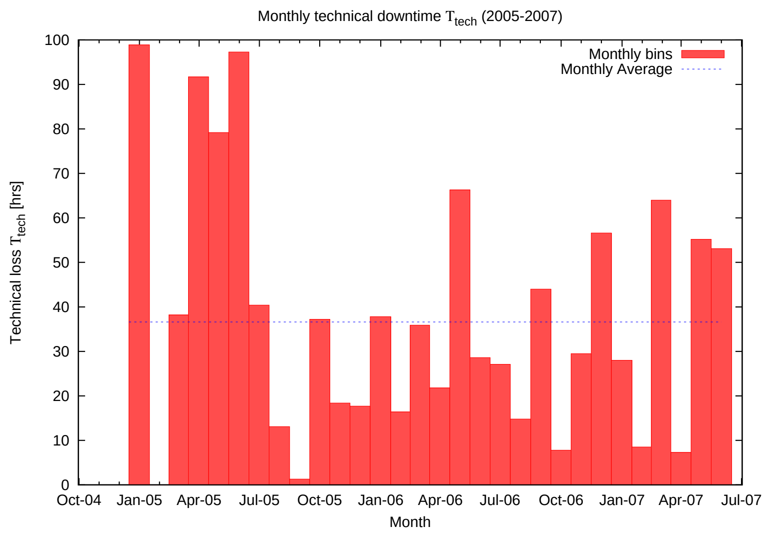


Figure 4.3: Monthly averaged technical downtime (hours) over the period 2005 - 2007 (30 months). The data only includes periods where there was no weather downtime. Average time per month lost was 36.6 hrs or 1.2 hours per night

realistically be achieved is to use the long term probabilities to estimate the likelihood of technical downtime over extended periods.

Disruption to power supplies are sometimes known in advance and could be factored into medium term planning. Periods of planned maintainance are often known well in advance and could also be factored in. Prior to distribution of the Phase 2 entry system to users, both of these were implicitly taken account of by the operations team when setting up the content of the Phase 2 ODB but this information was not known explicitly by the scheduler.

4.3 Weather

Weather statistics have been obtained from 3 independant sources.

- Data from the LT's own weather monitoring system (WMS) has been collected by embedded software over a 20 month period.
- Archived meteorological station data from various facilities at the ORM are available (Sorensen (2002)) going back to 2002.
- Observer reports of weather downtime hours per night (collated next day) for reporting on the telescope website. This data is bundled in with the technical downtime statistics and overrides these when both occur on the same night.

4.3.1 Telescope Weather Monitoring System

The Weather Monitoring System (WMS) provides feeds of various meteorological data from a weather station on site located 20m from the telescope enclosure. Data is provided to the RCS at a cadence of around 5 seconds. The RCS filters this data and uses a set of rules (Table. 4.1) to decide if the weather should be classified as *good* in which case (all things being otherwise okay) observing may proceed, or *bad* in which case observing may not proceed or if already underway should be stopped and the telescope and enclosure made safe.

Automated weather shutdowns based on WMS data are triggered by any of the following sources:-

- High humidity can lead to condensation on cold surfaces (mirror, electricals) and is an indicator of cloud and potential precipitation.
- Rain causes wetting of all equipment, this has to be avoided especially on sensitive optical, electrical and hydraulic systems.
- Moisture fraction is indicated by a digital sensor and indicates that rain or condensation is occuring or has recently occurred but not yet cleared sufficiently.
- Cold temperatures lead to ice which can cause the enclosure portals to stick and thus put considerable strain on the motors and electrical supply if an attempt was made to open these.
- Wind gusts can cause damage to telescope structure and attached instruments. Moderate wind can also lead to poor tracking performance due to *wind shake*.

Table 4.1 summarizes the rules in force (January 2008) for weather clear and alert triggers. Currently all rules have a 30 minute clearing stability time but this is a configurable parameter.

The data collected consists of 1582518 samples taken at around 30 seconds cadence over a period of 20 months between 2005 and 2007. The inter-sample gap distribution indicates that there are relatively few gaps. The vast majority of the samples occur with gaps of 30 seconds or less. A small number 292 exceed 5 minutes, a further 14 excede 30 minutes and 58 excede 60 minutes. The largest gap of 4.2 days occurred as a result of a site power outage

Weather variable triggering conditions				
Variable	Alert threshold	Primary clear level	Secondary clear level	Stability parameter
Humidity	$> 80\%$	$< 70\%$	$< 75\%$	30 min
Moisture	$> 10\%$	$< 9\%$	$< 9.5\%$	30 min
Wind speed	$> 15ms^{-1}$	$< 12ms^{-1}$	$< 14ms^{-1}$	30 min
Temperature	$< 0.0°$	$> 0.1°$C	$> 0.05°$C	30 min

Table 4.1: Definitions of alert and clear threshold levels for triggering good/bad weather conditions. A variable crossing its alert level signals bad weather. In order to clear (signal good weather) the variable must pass the primary clear level and remain below the secondary level for at least the time specified by the stability parameter. All variables must be in the *clear* state for overall *Good* weather. Any variable in its *alert* state indicates *bad* weather.

4.3.2 Analysis of results

The distribution of humidity is shown in Fig. 4.4. The distribution peaks around 15% and average of 40%. A very sharp secondary peak occurs around 95-100% humidity. With the variable trigger levels (Sect. 4.1) set to 70% (*clear*) and 80% (*alert*) these can both be seen to be well into the tail of the main distribution just before the spike. This suggests that few false alerts should occur.

The distribution of moisture fraction is shown in Fig. 4.5. The distribution is strongly peaked with average 0.07%. A total of 89% of measurements are below the alert level.

The wind speed distribution is shown in Fig. 4.6. The distibution is wide with average 5.37ms^{-1}. Only 2% of samples are above the alert level.

Fig. 4.7 shows the distribution of external temperature at the site. The average temperature at the site is 9.43°C. Only 3.7% of time is the temperature below the alert level.

From (Table 4.2) which shows the relative fractions of time when each of the weather triggers is within the alert region, it is clear that humidity is the greatest contributor to bad weather. The next highest is moisture level but these are quite closely

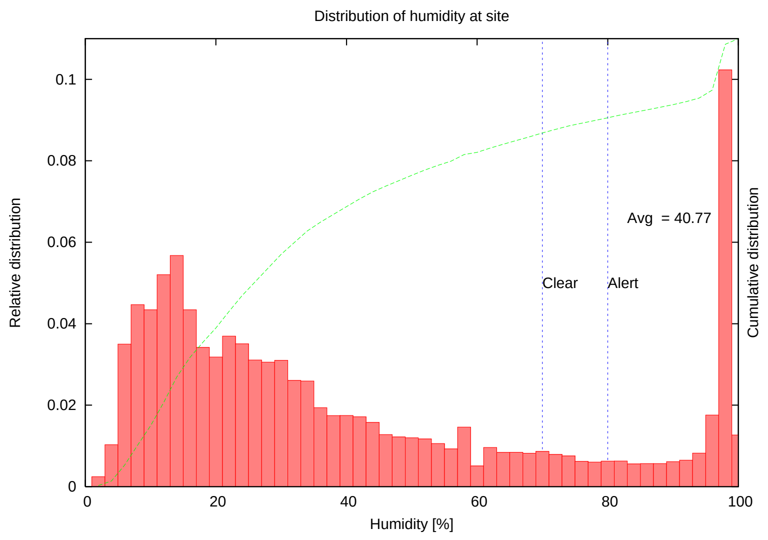


Figure 4.4: Distribution of atmospheric humidity at the telescope site over all samples. The distribution shows a profile with main peak around 15% and average of 40%. A very sharp secondary peak occurs around 95-100% humidity. With the variable trigger levels set to 70% good and 80% bad these are in the tail of the distribution just before the spike level.

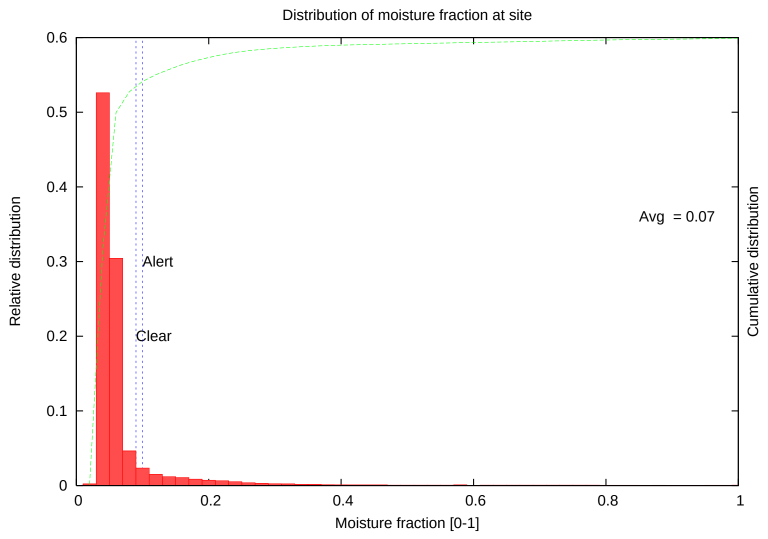


Figure 4.5: Distribution of moisture fraction at telescope site. Some 89% of nights were below the alert threshold.

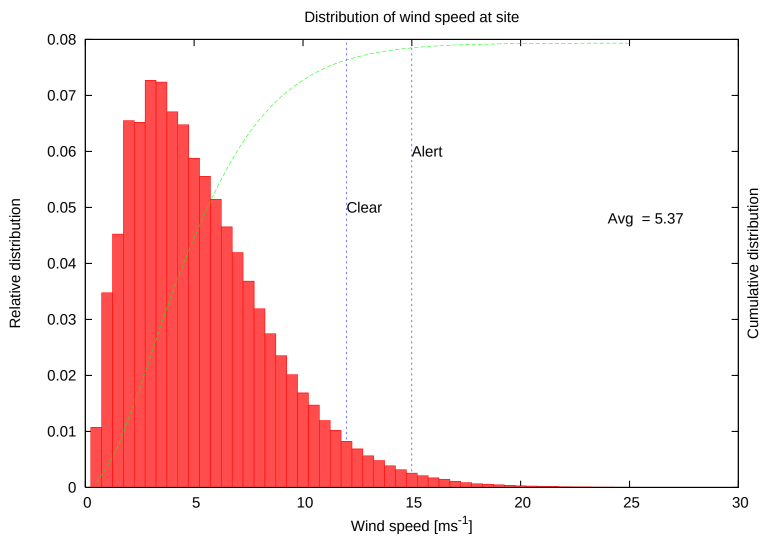


Figure 4.6: Distribution of wind speed at telescope site. Only 2% of nights were above the alert threshold.

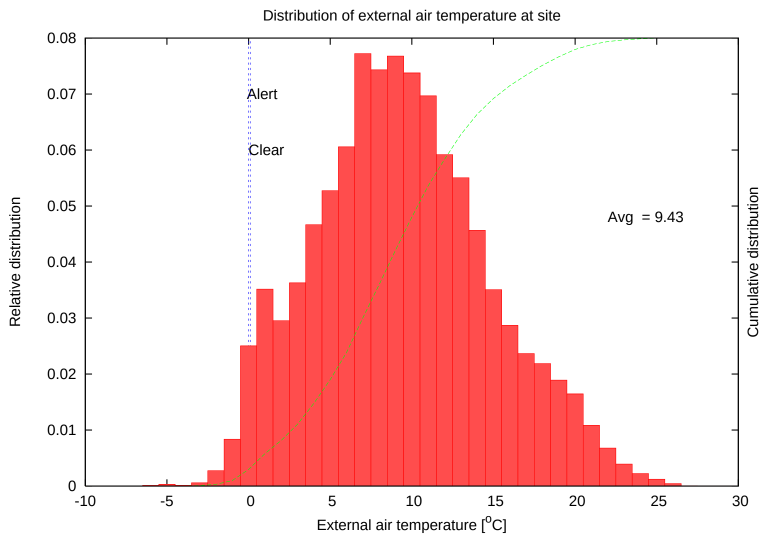


Figure 4.7: Distribution of temperature at telescope site. Average temperature over the year is $9.43°$C with only 3.7% of nights below the alert (freezing) temperature.

related. Consequently it is reasonable to use humidity alone for any prediction studies.

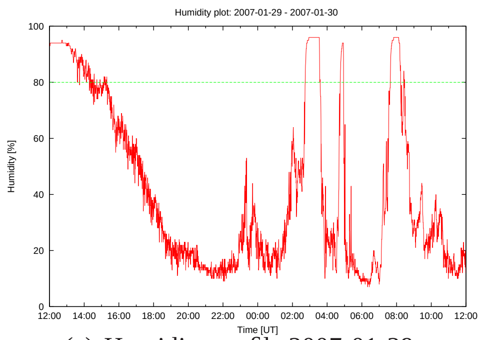
Fraction of weather variable above alert trigger level	
Variable	Fraction above alert level
Humidity	18%
Moisture	11%
Wind speed	2%
Temperature	3.7%

Table 4.2: Fraction of recorded weather variable statistics over *alert* level. Humidity is the largest contributor at 18%.

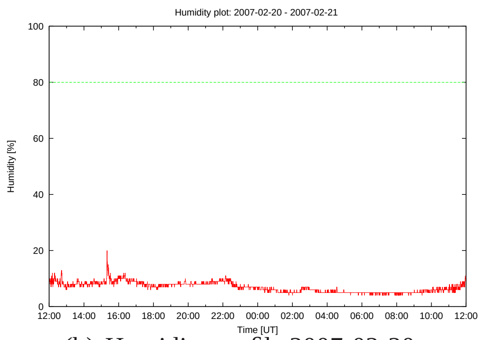
Some examples of humidity profiles are presented in Figs. 4.8. It can be seen that on these individual nights there can be several periods of good and bad weather making forward planning difficult. Software was written to analyse the recorded data and generate statistics of good and bad weather periods based on the rules in Table. 4.1.

The results of this analysis are presented in Fig. 4.9 and shows the relative fraction of *good* weather over various sized time bins. Bearing in mind the small period of data available (in climatological sense) there appears to be a tendency for better weather in summer (May-July) with increasingly higher fraction of bad weather in winter in accord with general observations.

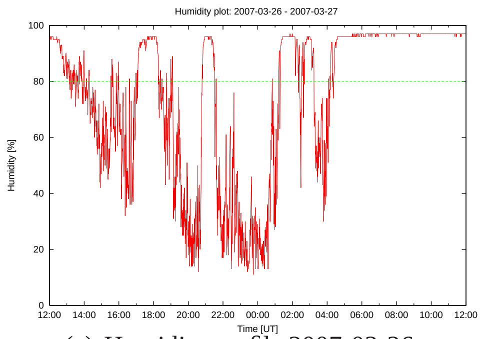
Further analysis was performed on the WMS data to yield the distribution of lengths of good and bad weather periods. Fig. 4.10 shows the variation (i.e. number of periods) of lengths of consecutive periods of good and bad weather based on humidity threshold (80% trigger) and clearing stability time of 30 minutes. We see that shorter periods are most common with the average length of *good* periods being 31.5 hours and bad periods being 7.7 hours. Cumulative plots are shown in Fig. 4.11. The overall fraction of time classified as *good* resulted in 80.03% of all time with *bad* weather making up the remainder. A rapid drop off suggests that long periods of continuous good/bad weather are rare, however outliers make this awkward as a source for prediction.



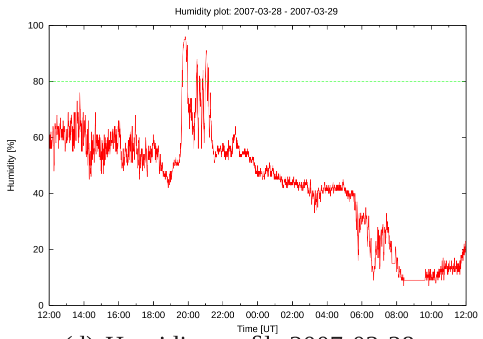
(a) Humidity profile 2007-01-29.



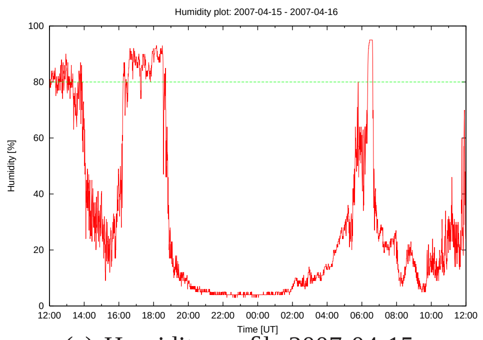
(b) Humidity profile 2007-02-20.



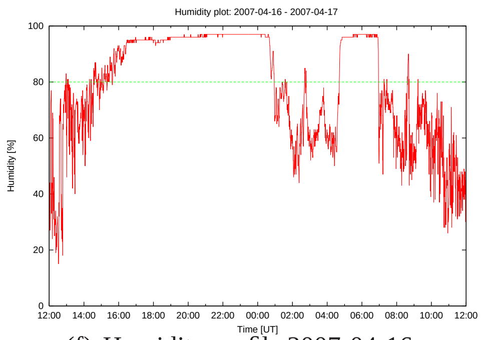
(c) Humidity profile 2007-03-26.



(d) Humidity profile 2007-03-28.



(e) Humidity profile 2007-04-15.



(f) Humidity profile 2007-04-16.

Figure 4.8: Example humidity profiles. As can be seen there is some considerable variation from night to night. Humidity can suddenly rise from what looks like a stable low level through the alert level in avery short period of time. This is often due to cloud spilling over the rim of the caldera.

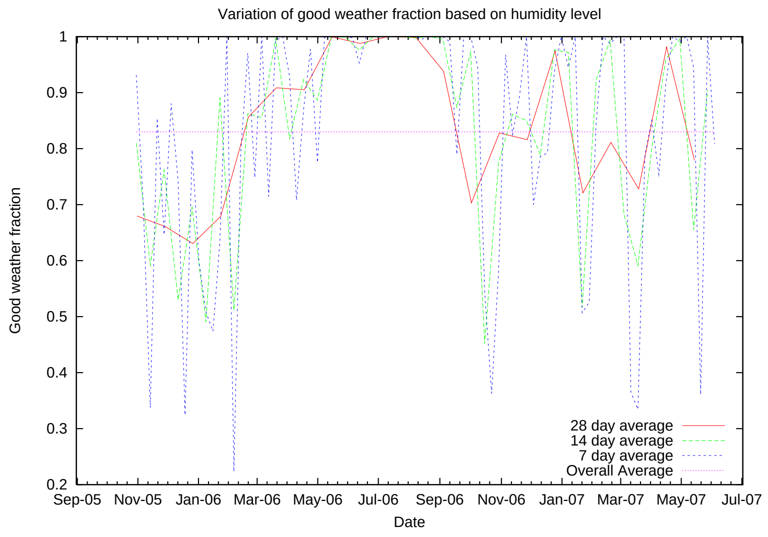


Figure 4.9: Monthly averaged good weather fraction ($1 - \Delta_W$) over the period 2005-2007 (20 months) based on WMS humidity levels averaged with various bin sizes. There is considerable month to month variation but summer generally has the highest fraction of good weather.

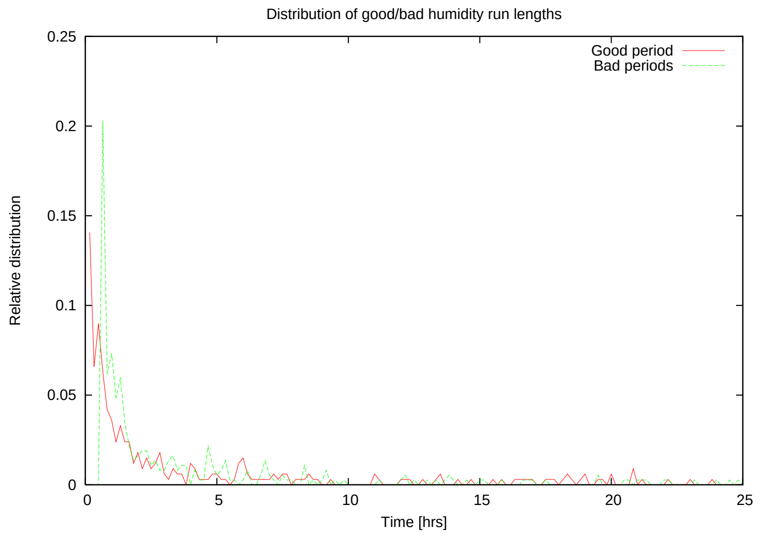


Figure 4.10: Relative probability of lengths of good/bad weather runs. Though around half the continuous periods of good or bad weather are less than 1 hour, there is a considerable tail in the distribution.

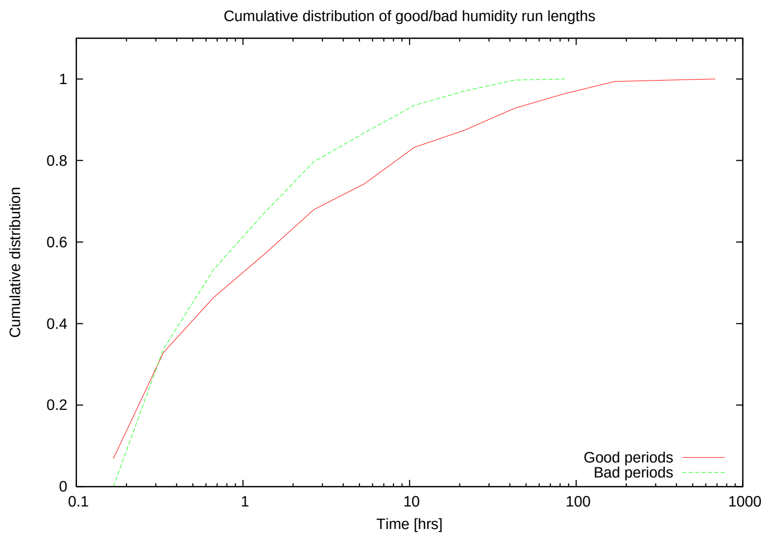


Figure 4.11: Cumulative probability of lengths of good/bad weather runs. Only 5% of bad runs last more than 10 hours while some 20% of good runs last 10 hours or more.

4.3.3 Prediction experiment

If we wish to factor weather statistics into the planning and scheduling decision making processes, then without additional information the best we can do is to use the long-term climatological prediction. Based on the data plotted in Fig. 4.12 showing the lengths of periods of continuous *good* and *bad* weather, a simple prediction model was tested based on the assumption that the weather will continue in its current state for a period roughly equal to the length of time it has already been in that state continuously, thereafter the probability of maintaining this state would decrease at an assumed exponential rate with a decay length some multiple of the current stability period. As an example, assume the weather has been *good* for about τ hours. We then assume it will continue to remain *good* for about another τ hours with the probability decaying as $e^{-t/m\tau}$, m being a scale factor yet to be determined.

A set of simulations were run using the extracted period data (Fig. 4.12) to determine the effectiveness of this prediction mechanism. Every 15 minutes through the available period a determination is made of the current weather state and how long (τ_G *good*) or (τ_B *bad*) it has been in that state. A prediction is made at a number steps into the future (192 steps of 15 minutes constituting up to 48 hours look-ahead) using

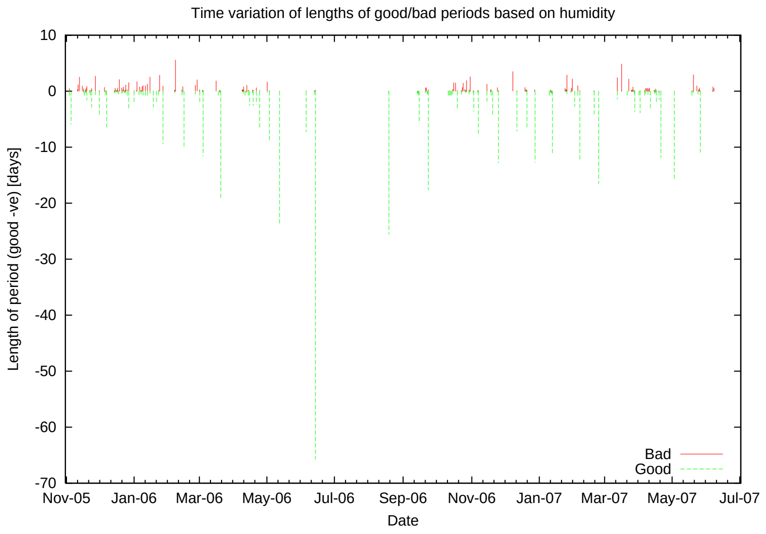


Figure 4.12: Time variation of lengths of good/bad periods based on humidity level. There are more longer periods of good weather than bad. This information was used to setup an experiment to test the ability to predict weather based on the length of time in a particular state using Eqn. 14.

the rule specified in Eq. (14). At each step the prediction is compared to the actual weather state at the time and counted as either a hit (correct prediction) or miss (incorrect prediction). The final percentages shown in Fig. 4.13 against look-ahead time for a number of decay scale factors m. The baseline of 80.03% represents the worst we should be able to acheive on average based on long-term climatological prediction - basically if we always just guess that the weather will be good this will work 80.03% of the time. Fig. 4.14 shows the crossover point - the length of look-ahead where the prediction becomes worse than long-term climatological prediction as a function of the decay scale factor m. This is seen to converge towards a value of 30-31 hours which is close to the average length of *good* weather period.

$$P_{good}(\Delta T) = \begin{cases} \Delta T < \tau_G : & 1 \\ \Delta T > \tau_G, \tau_G < T_G : & e^{\frac{\Delta T - \tau_G}{m\tau_G}} \\ \Delta T > \tau_G, \tau_G > T_G : & e^{\frac{\Delta T - T_G}{m\tau_G - T_G}} \end{cases} \quad (14)$$

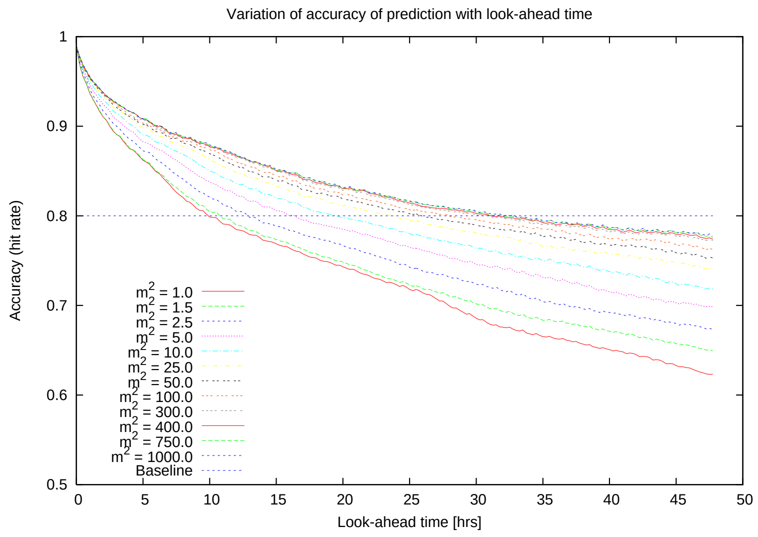


Figure 4.13: Effect of time scaling-factor (m) on variation of prediction accuracy of time decay prediction model against look-ahead time. All plots show a decay with time. As the scale factor is increased the cross-over point for the prediction against climatological baseline prediction of 80.03% approaches a maximum of around 31 hours.

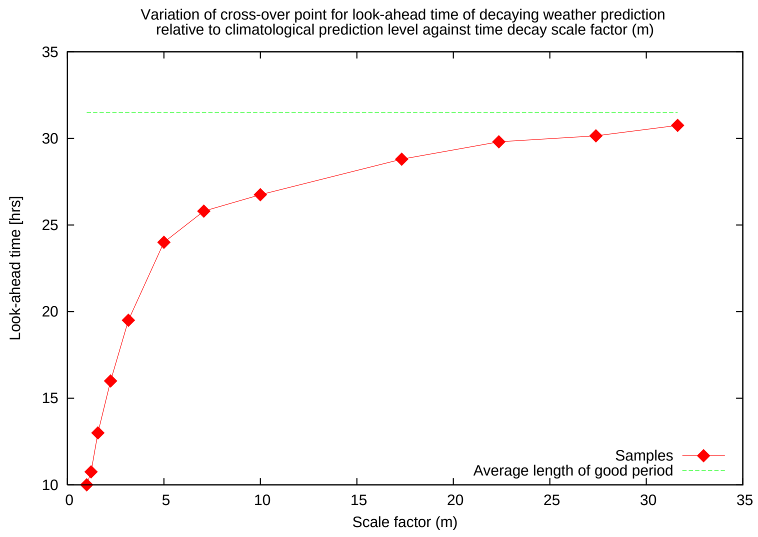


Figure 4.14: Variation of cross-over point for look-ahead weather prediction using time decay model with scale factor.

4.3.4 Observer reports

The observer-reported hours-per-night data for weather downtime are displayed in figures 16(a) for 2005 , 16(b) for 2006 and 16(c) for 2007 (part).

Figure (4.15) shows the variation of weather downtime fraction, the fraction of the potential observing hours per night lost to bad weather (Δ_W) averaged by month for the available data. June is the best month with less than 10% of potential observing time lost to weather. February is worst with 58% of potential observing time lost to weather.

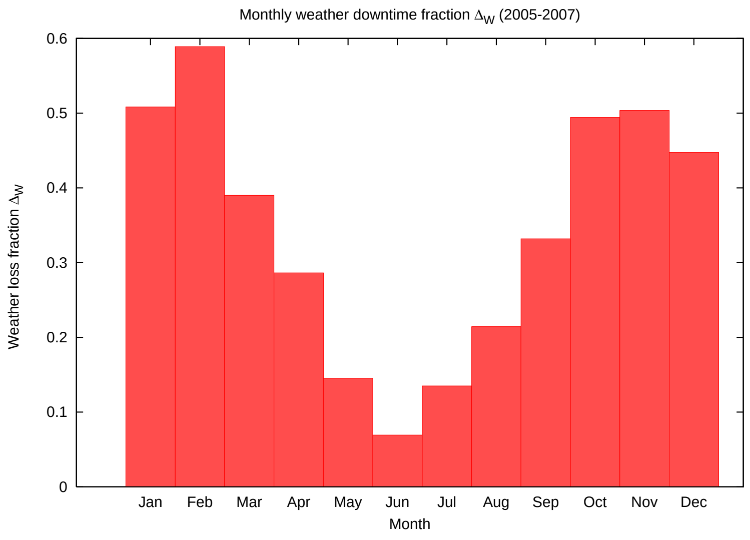
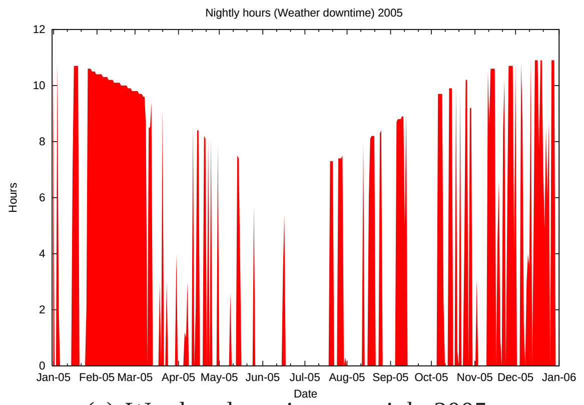


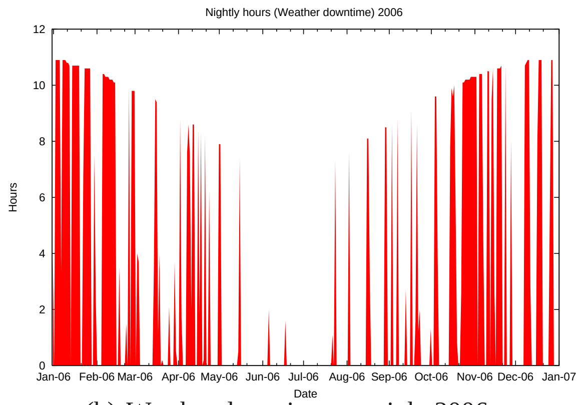
Figure 4.15: Monthly averaged weather downtime fraction over the period 2005 - 2007 (30 months).

4.3.5 Analysis

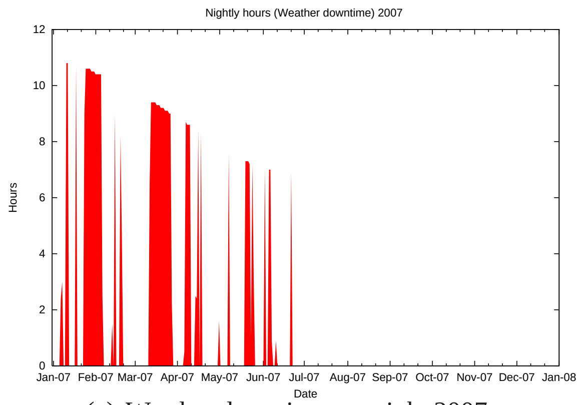
This data is subject to human interpretation. There is no hour by hour detail, only nightly totals. There is also no combination data (when bad weather and technical downtime occur simultaneously). Additionally there is a bias such that when combinations do occur, this is logged as bad weather. The plots do reveal a tendency for better weather in summer and more bad weather in winter as might be expected but beyond that little of use in prediction. Plots of run lengths, consecutive days where the



(a) Weather downtime per night 2005.



(b) Weather downtime per night 2006.



(c) Weather downtime per night 2007.

Figure 4.16: Nightly plots for bad weather periods (hours) for years 2005, 2006, 2007(part). There is clearly more good weather in the summer months.

fraction of time lost to bad weather exceed given thresholds, are shown in Figure 4.17. We see that shorter runs of good and bad weather are more common

Using this plot one can predict the likely length of a current run of bad weather based on the length up to the present time using Bayes theorem E.g. if the current run is 2 days long so far, the probability of the run going on for another 12 days or more is P(14)/P(2) = 0.23.

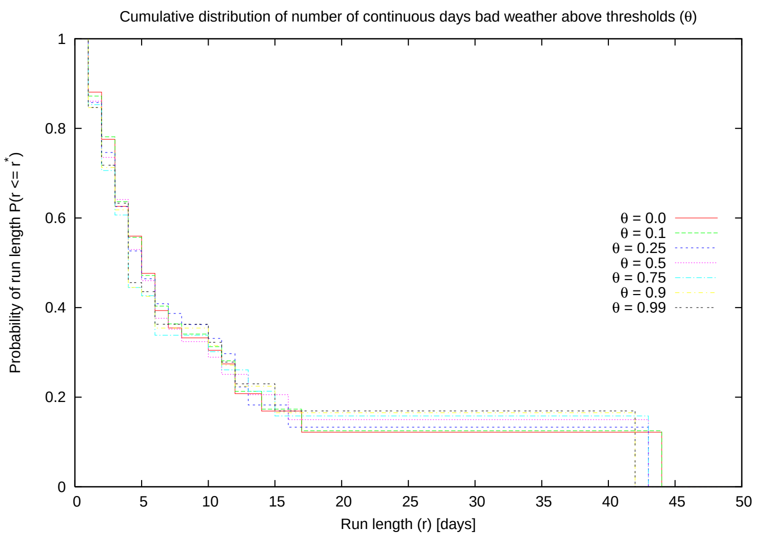


Figure 4.17: Probability of the length of a run of continuous bad weather for bad weather fraction (Δ_W) exceding threshold values between 0 and 1. Using this plot it is possible to determine the the likely length of the current bad weather run based on the length of run to-date.

4.3.6 Conclusions

Meteorological information obtained from logs of the telescope's own weather monitoring system (WMS) and from nightly observing logs was analyses and it was found to be suitable for observing for about 80% of the time averaged over the year. There is generally more good weather in summer than winter with $< 5\%$ bad weather in June rising to as much as 60% in February. The major contribution to bad weather is high humidity accounting for around 90% of such time. Other factors such as high winds and freezing temperatures account for only a few percent of the bad weather.

Using an analysis of the distribution of lengths of good and bad periods, a simple prediction model was designed and tested against recorded data. It was found capable of anticipating the length of the current period of weather with a degree of accuracy exceeding the long-term average for up to 30 hours ahead though the accuracy decreased with look-ahead time.

Further analysis shows that in runs of continuous bad weather around 50% of bad weather runs are < 5 days long with < 10% of runs exceeding 15 days. This information could be useful in longer-horizon planning.

4.4 Atmospheric Seeing and Extinction

The atmospheric seeing regime has been studied for many years both through short campaigns and as part of longer studies aimed at site testing for telescopes such as the Grande Telescopio Canarias (GTC). A short study in 1990 by Vernin and Munoz-Tunon (1992) using equipment at the Nordic Optical Telescope site (NOT) on the contributions of the various air layers above the site suggests that the main contribution, some 50%, comes from air in the boundary layer (from a few meters to 1 km above the ground. They found that 40% is provided by the free air above 1 km and that 8% comes from the surface layer of the first few meters. Sources internal to the dome provide the remaining 2%.

A longer 9 month study (Muñoz-Tuñón et al., 1997) using a DIMM mounted on a 5m tower (above the surface layer) concluded that the inversion layer which lies below the observatory site is of particular importance in determining seeing characteristics. The layer generally lies between 1200m and 1500m, well below the site at 2400m and acts to suppress convection - a layer of strato-cumulus is generally seen at the top of this layer. Around 55% of the local atmospheric humidity is trapped below the layer with around 20% above. They find that the best seeing correlates with those times when the inversion layer is at its lowest (around 1200m) and strongest (largest temperature difference between top and bottom of the layer) which occurs in the summer months

(June, July and August). At this time the strength of the trade winds is highest. They find that during the summer the average seeing is 0.61” and median 0.5”. During the remaining months the average seeing is 0.77” and median 0.91”.

Muñoz-Tuñón et al. (1997) also studied some time variation effects finding that typically seeing can change very abruptly, deteriorating within a few minutes but that it rarely returns to a stable level so quickly. Typically they find a recovery time of around 2 hours which they put this down to the effects of sudden perturbations in a steady atmospheric flow giving rise to turbulence which can take some time to settle. Some oscillatory effects were recorded with a period of around 45 minutes.

A year long study by Muñoz-Tuñón et al. (1998) on the variation of seeing between different sites at the ORM revealed an average seeing of 0.72” and median 0.65”. They found relatively small variation between sites except when the seeing was particularly poor when the variation was more pronounced - up to 0.2” between locations. In summer they find that seeing is better than 0.5” for 50% of the time dropping to 25% averaged over the whole year. A correlation was found between wind speed and direction and seeing quality.

At low wind speed ($<$ 5 km/h) there was no relation between wind direction and seeing. At medium wind speed (5 - 15 km/h) they found the best seeing associated with Northerly winds and poor seeing when the wind was Southerly (from over the Caldera rim). At high wind speeds ($>$ 45km) seeing was generally very poor.

In a further study of site conditions, Vernin and Muñoz-Tuñón (1998) find a relaxation time for seeing to return to *normal* after an excursion to be around 1.2 hours.

4.4.1 Collected seeing data from archived images

Data from the LT image archive, originally extracted from FITS headers was collated and processed to give seeing statistics. The processing included correcting for target zenith distance z and wavelength λ (Eq. 15)[5] using details of the instrument filter choice, correction for binning and removal of outliers caused by:- (*i*) Some images are deliberately defocussed or telescope out of focus, (*ii*) Images of extended sources cause problems for reduction pipeline, (*iii*) Other general pipeline problems.

$$s(z=0,\lambda_R) = s(z,\lambda)(\frac{\lambda}{\lambda_R})^{-0.2}\sec^{0.6} z \tag{15}$$

The number of images available for processing per month are shown in Fig. 4.18. There are at least 2000 images per month, in some cases up to 11000. On average around 5000 giving a worst case statistical noise of around $\pm 2.2\%$.

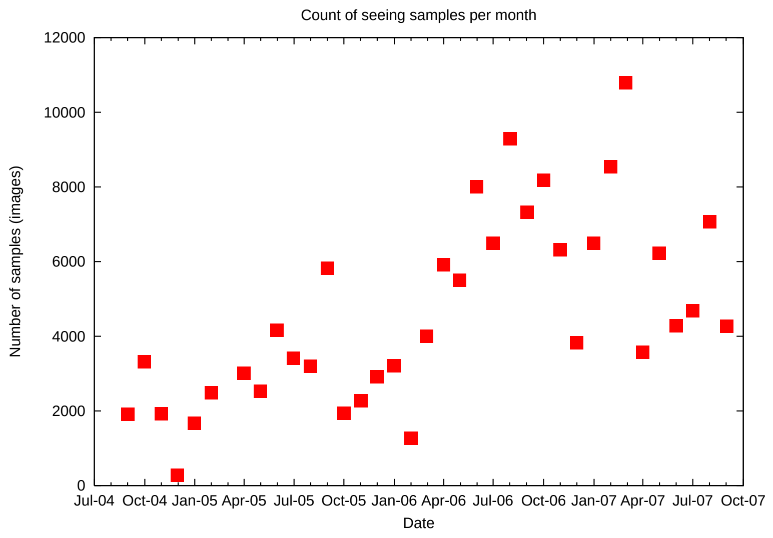


Figure 4.18: Monthly count of images used for deriving seeing statistics. There are at least 2000 samples available on each of the months with a maximum of 11000.

Figures 4.19 and 4.20 show the relative and cumulative distributions of atmospheric seeing over the full period of available images both raw and corrected for elevation and wavelength. Table 4.3 shows the quartiles of seeing data.

[5] The correction of seeing to R-band (λ_R) at zenith is due to work by Sarazin and Roddier (1990) based on earlier work by Fried (1966)

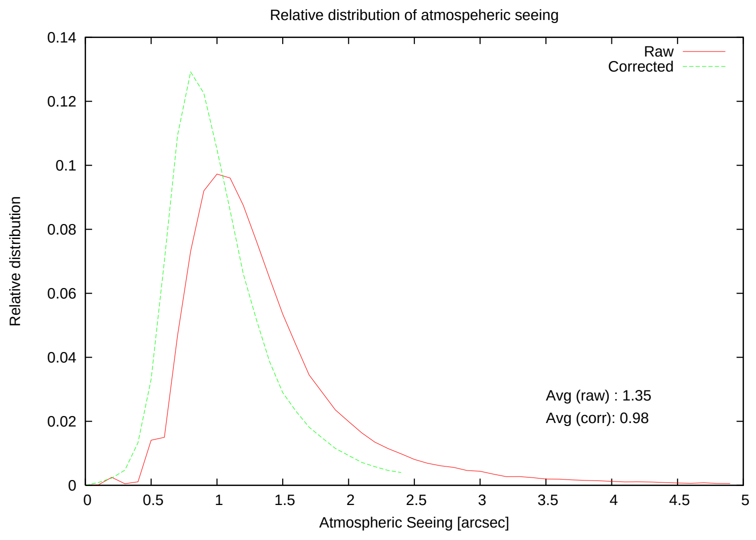


Figure 4.19: Relative distribution of r-band seeing data. For raw images the average seeing is 1.35”while for corrected images this reduces to 0.98”. Both distributions are broad with 50% of raw images between 0.97” and 1.63” while for corrected images 50% of images lie between 0.8” and 1.3”.

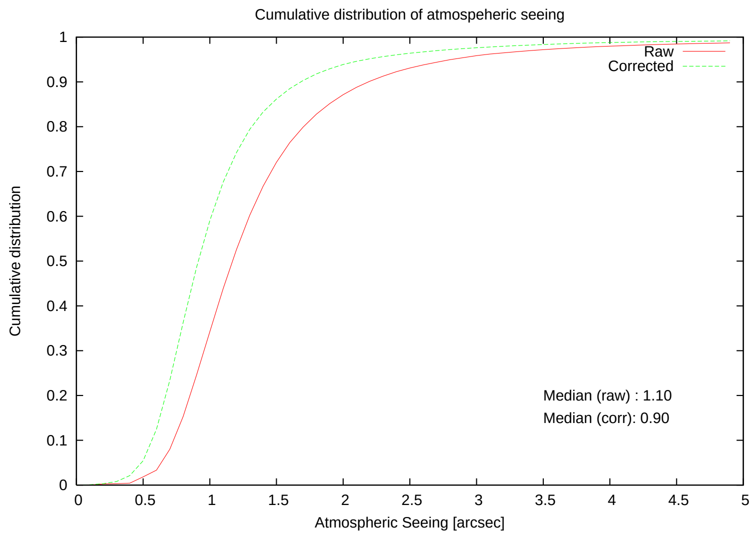


Figure 4.20: Cumulative distribution of seeing data.

Seeing quartiles (r-band)		
Quartile	Raw	Corrected
Q1	0.97”	0.8”
Q2	1.25”	0.95”
Q3	1.63”	1.3”

Table 4.3: Quartiles of raw and corrected seeing distributions extracted from Fig. 4.19

4.4.2 Variation of seeing during the night.

It is a common belief that there is a systematic variation in seeing quality during the night, This has been found to be incorrect by Muñoz-Tuñón et al. (1997). The data collected and displayed in Fig.4.21 shows the variation of average seeing with time (binned by UT) over a 3 year period. Allowing for the variation of a given UT time with *time after sunset* over the year there does not appear to be a large variation over the darker part of the night. Fig. 4.22 shows the relative number of samples per UT bin, typically around 1500 samples per bin indicating low levels of noise (around $\pm 2.5\%$). Fig. 4.23 shows some examples of nightly seeing profiles. On any individual night the seeing may increase, decrease, remain relatively stable or vary quite dramatically.

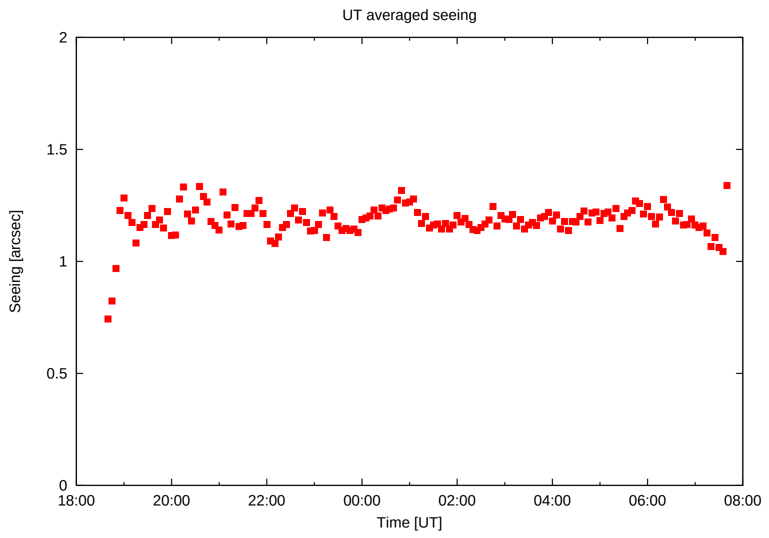


Figure 4.21: Seeing averaged by UT binning of time over all available nights. There is clearly no systematic variation over the course of the night.

4.4.3 Monthly variation of seeing.

Fig. 4.24 shows the variation of r-band seeing averaged per month over the set of available images. Seeing appears to be better over the summer months (typically around 1.0”), deteriorating markedly during winter to around 1.5” in agreement with Muñoz-Tuñón et al. (1997) who find the best seeing from June to August.

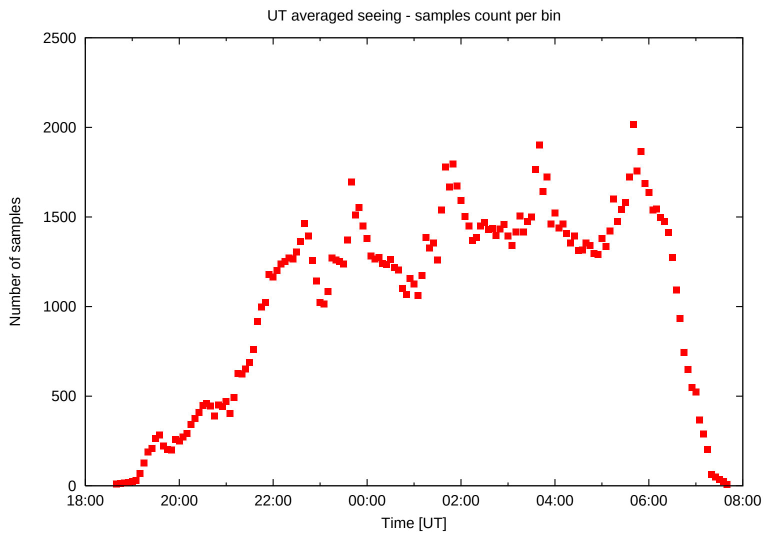
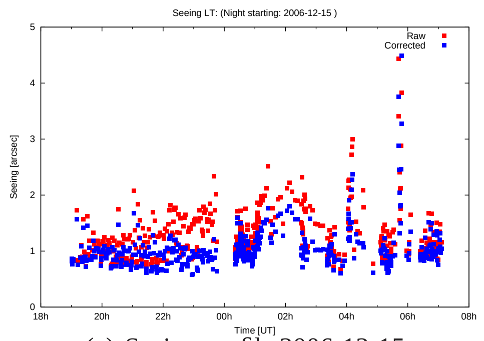
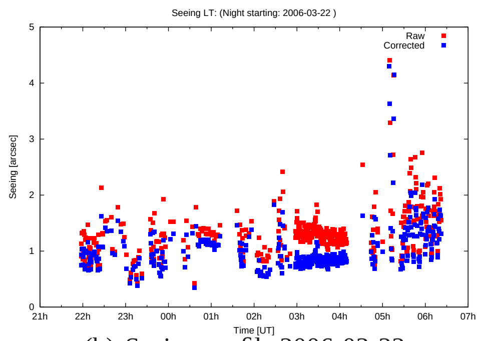


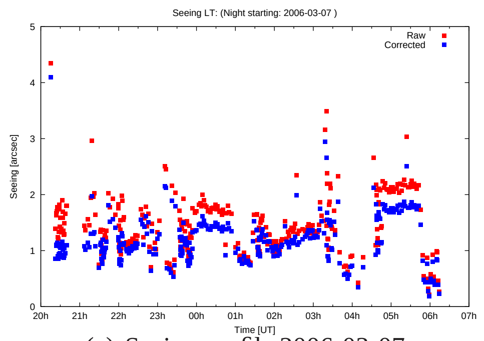
Figure 4.22: Counts of samples per bin for UT averaged seeing data. The large number of samples, typically around 1500 for each bin suggests only very low levels of statistical noise (typically ± 0.05") will be present in the data.



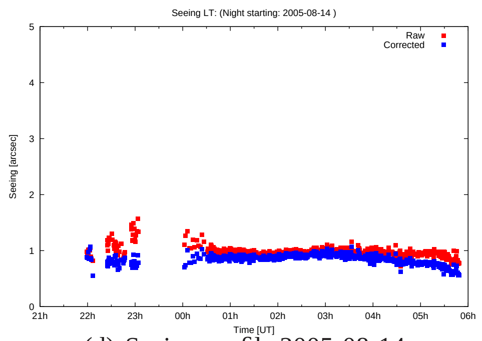
(a) Seeing profile 2006-12-15



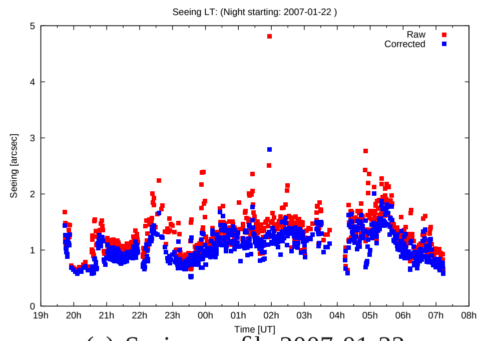
(b) Seeing profile 2006-03-22



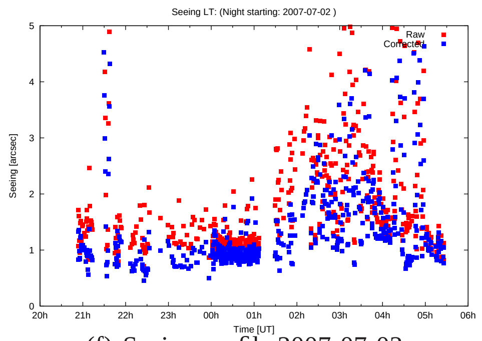
(c) Seeing profile 2006-03-07



(d) Seeing profile 2005-08-14



(e) Seeing profile 2007-01-22



(f) Seeing profile 2007-07-02

Figure 4.23: Example r-band seeing profiles. As can be seen there is some considerable variation from night to night.

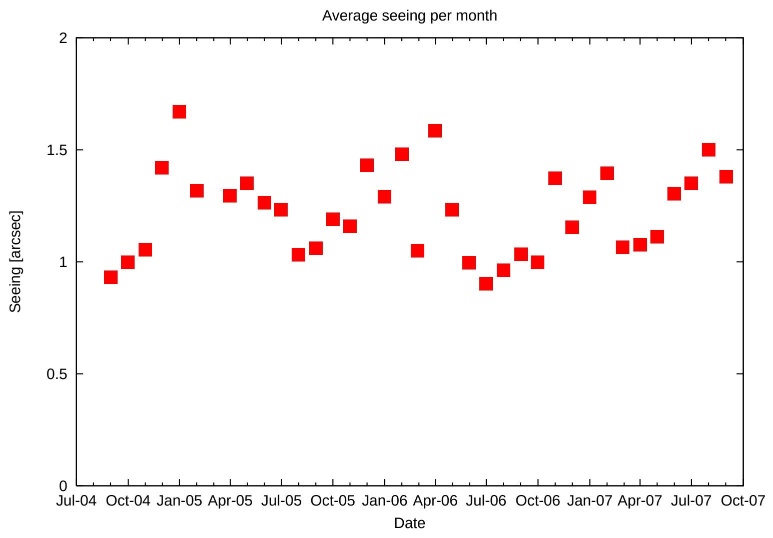
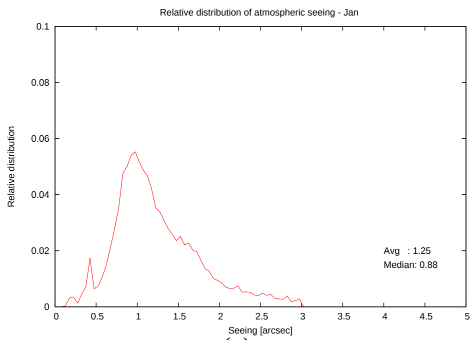
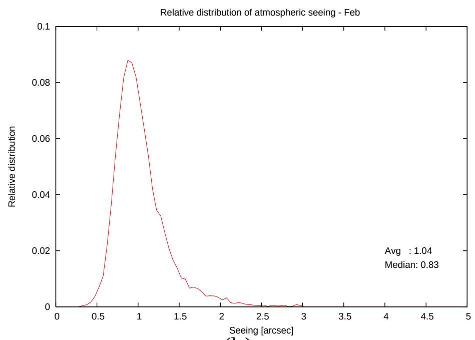


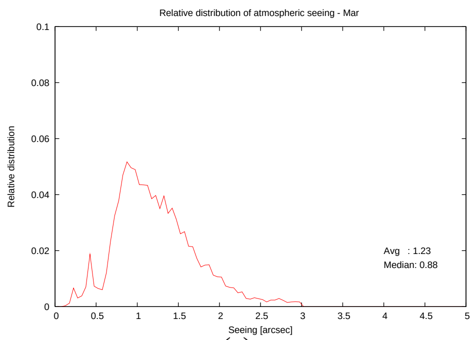
Figure 4.24: Corrected seeing averaged per month over all available images. The best seeing is during the summer months (roughly July to September) in agreement with Muñoz-Tuñón et al. (1997) who find the best seeing from June to August.



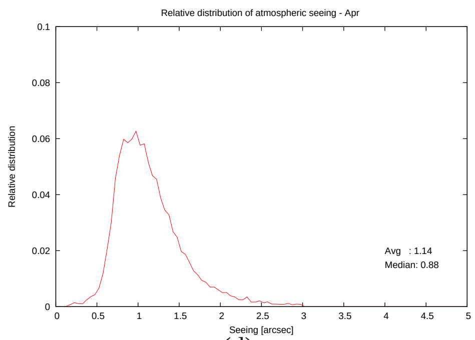
(a)



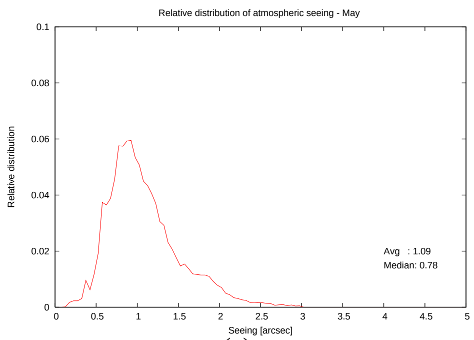
(b)



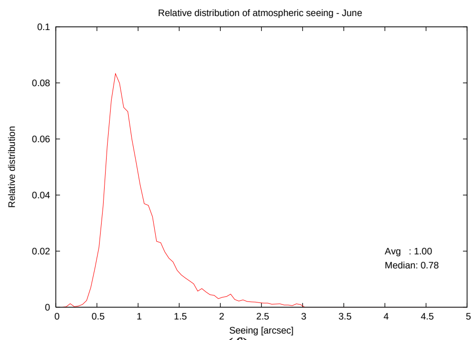
(c)



(d)



(e)



(f)

Figure 4.25: Relative r-band seeing distributions by month (January - June). The best seeing occurs in June and surprisingly February.

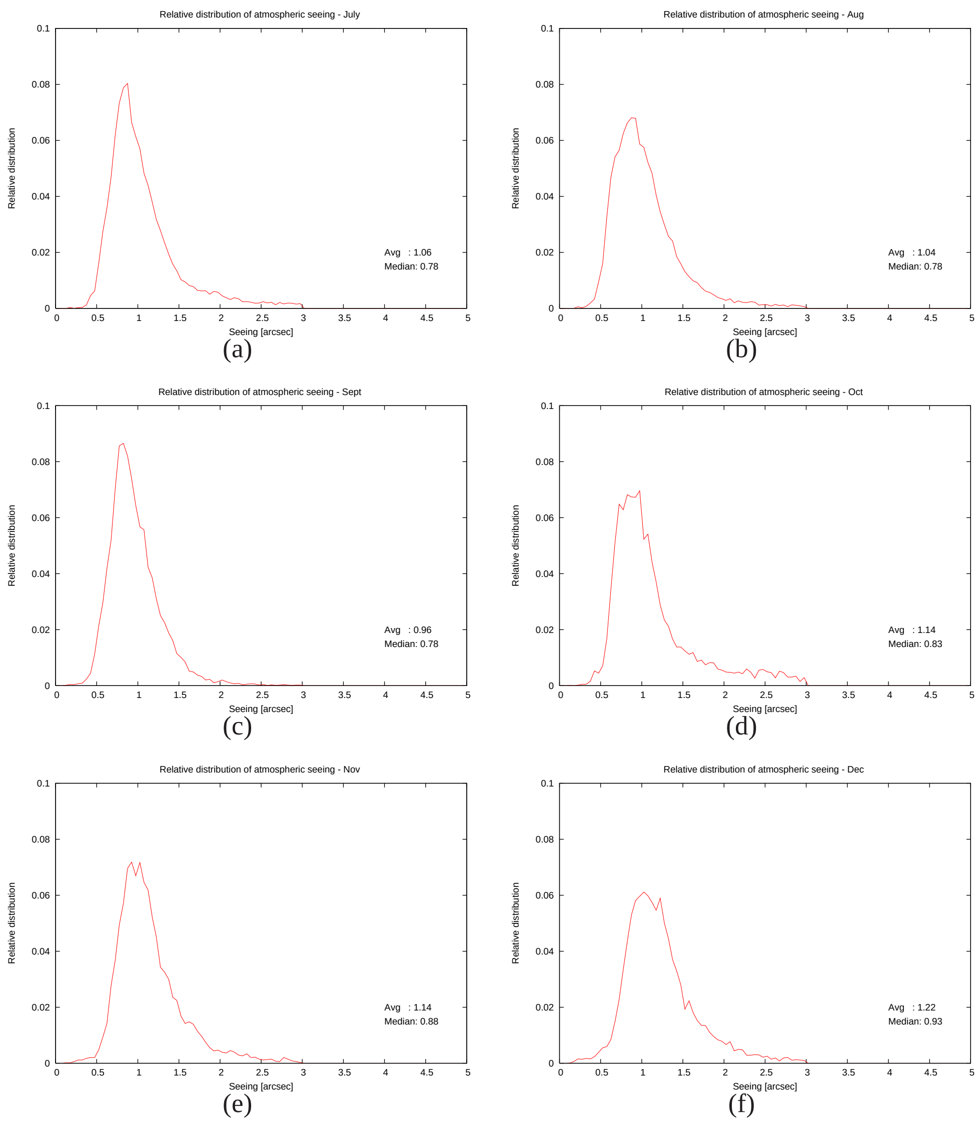


Figure 4.26: Relative r-band seeing distributions by month (July - December). Seeing deteriorates from October through December.

4.4.4 Extinction

A study of the contributions to extinction at the ORM was made by King (1985). This gives formulae for the calculation of the contributions from Rayleigh scattering and absorption by ozone and water-vapour. They find that the contribution from ozone can vary significantly during the year and even on timescales of a few hours.

A major contribution to extinction is the dust (calima) contained in the Saharan Air Layer (SAL). This dust is thrown up from the Saharan desert by predominatly South

Easterly winds and then pushed West over the Canaries and Atlantic, often reaching as far as South America and the Caribbean. In a study comparing CAMC extinction measurements with data derived from satellites Varela et al. (2007) find that during summer around 75% of nights are dust-free but during the rest of the year this rises to around 90%. Episodes of calima can however occur sporadically at other times. They find that k_V is typically $< 0.2 mag/airmass$ for 88% of times and $> 0.5 mag/airmass$ for 1% of nights with a modal value of 0.11 during dust-free nights. The cross-over between photometric and spectroscopic conditions occurs at $k_V = 0.153 mag/airmass$

A study of 2850 nights of CAMC data by Guerrero et al. (1998) reveals that most dust is present during June to September (coincidentaly the period of best seeing), they also note large changes in mean extinction during 1991 and 1982 corresponding to eruptions of Mt. Pinatubo and El Chichon.

Currently there are no automated means of providing extinction information to the scheduler though this is an observing constraint available to users. The information is entered by an observer early in the night based on a variety of indicators including:- (*i*) inspection of webcam images from LT and Nordic Optical Telescope (NOT), (*ii*) calima and cloud indications on satellite images, (*iii*) stability of insolation measurements during the day, (*iv*) variability of cloud temperature measurements from Boltwood Cloud Sensor (BCS) (Marchant et al., 2008) .

4.4.5 Sky brightness

In a detailed analysis based on 427 observations made with the Isaac Newton Telescope (INT) and the Jacobus Kaptyn Telescope (JKT) on La Palma between 1987 and 1996, Benn and Ellison (1998) [6] found that the sky background at the ORM is composed of contributions from (in decreasing order of precedence):- (*i*) Airglow, (*ii*) Zodiacal light, (*iii*) Stars (with $V > 20$), (*iv*) Starlight scattered by interstellar dust, (*v*) Extragalactic light, (*vi*) Light pollution.

[6] A regularly updated online version of this paper is available at: http://www.ing.iac.es/Astronomy/observing/conditions/skybr/skybr.html

They find the relative contribution from airglow and zodiacal light to be around 2.5:1 at high ecliptic latitude while at lower latitude the sky is brighter by 0.4 mag. The mean brightness across the sky does not vary by more than 0.1 mag between times of astronomical twilight. They present a formula for calculation of the sky brightness in V as a function of sky position in moonless conditions. A detailed study of the sky-brightness under moonlight conditions is presented in Krisciunas and Schaefer (1991).

Sky brightness is not currently used by the scheduler (other than via a lunar-elevation constraint which indicates *bright* or *dark* sky) but could be included as an additonal observing constraint if a suitable means of determining this were feasible.

4.4.6 Conclusions

Details of the atmospheric (R-band) seeing, measured by a real-time pipeline operating on data from the telescope's main imaging camera were extracted from the data archive and reduced taking into account pixel-scale, binning, filter wavelength and target elevation above the horizon. The results indicated that the median seeing on site is around 0.95". It was also noted that median seeing is best in summer, typically around 0.78" with average of 1.0" and poorest in December with a median value around 0.93" and average over the winter months of 1.5". It was further found that typically, and contrary to popular belief, no systematic variation in seeing quality occurs over the course of the night.

4.5 Phase 2 population

The Phase 2 ODB contains a large amount of data. There are 41 tables in the database and among these there are:- 16000 groups (around 500-700 active at any time) in 90 proposals belonging to up to 200 users. There are a total of $>$ 100000 observation sequence elements (instructions on what to do) of which around 5000 are active at any time. Over the last 2 years of operation of the current database implementation (August

2009 to July 2011) some 20000 execution history elements, recording the completion of group executions have been inserted.

As examples of the numerous statistics which could be extracted from the ODB to characterize its content, Fig. 4.27 shows the distribution of lengths of monitoring periods for groups with *monitor* or *minimum interval* timing constraints while Fig. 4.28 shows the distribution of lengths of observing sequences (based on exposure lengths) for groups in the ODB. As can be seen from both these plots there is a wide range in these characteristics along with some detailed structure. This gives at least a flavour of the complexity of trying to characterize the ODB content.

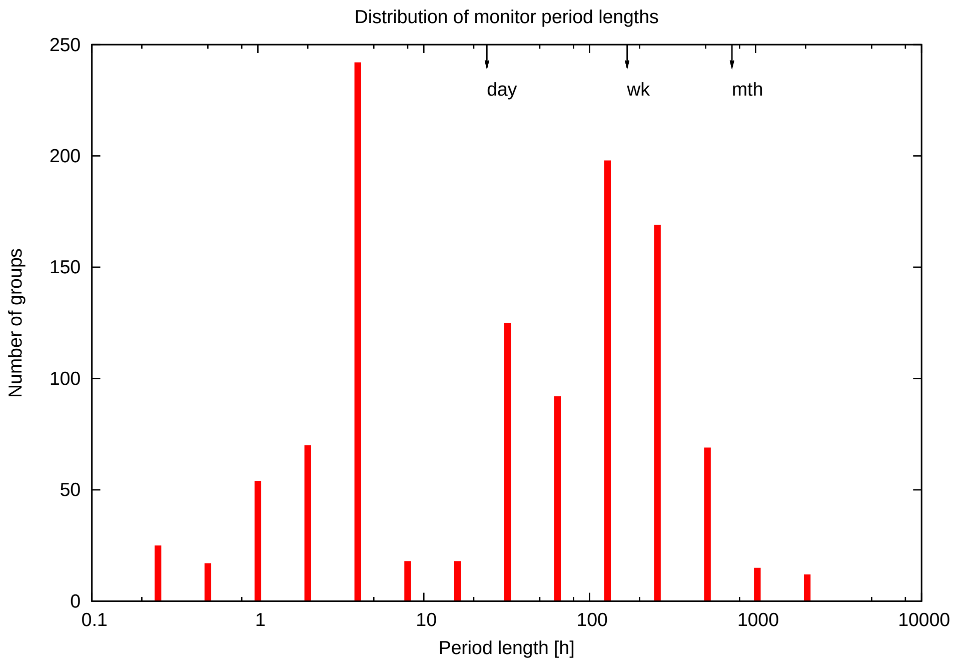


Figure 4.27: Distribution of lengths of monitoring periods for repeating groups over full ODB content. There are peaks at 4 hours, 1 day and 1-4 weeks.

As an example of how this complexity manifests itself in the scheduling process, Table. 4.4 shows the scores for the set of feasible candidate groups on a particular despatch scheduler sweep in rank order.

The scores plotted in Fig. 4.29 show that the top ranked group in this case *J0129* is well ahead of any rivals, however the lower ranked groups are closer together. Over the course of the night the scores of the winning groups are shown in Fig. 4.30. It

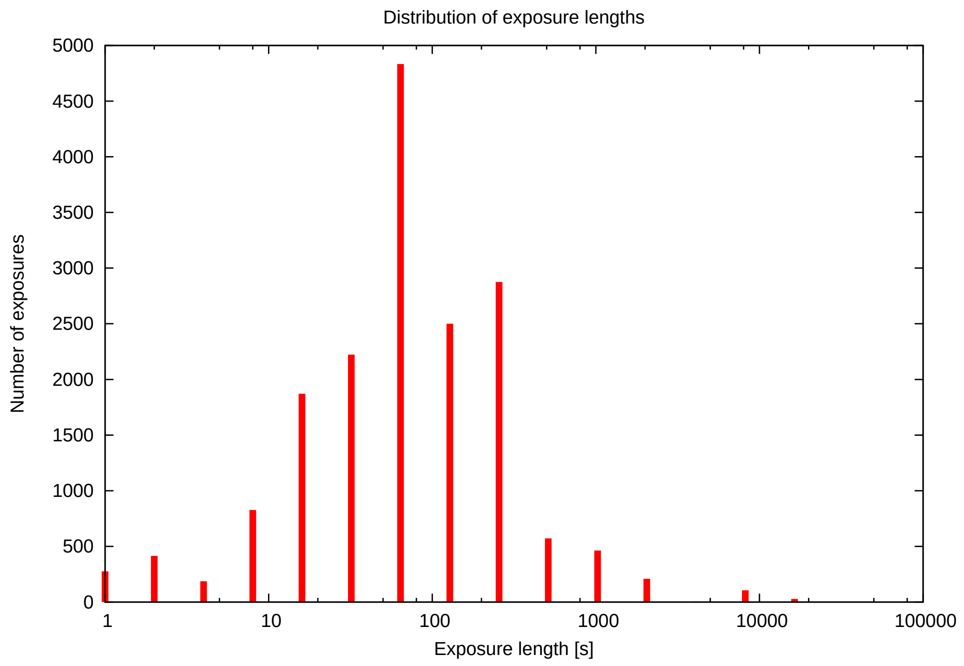


Figure 4.28: Distribution of lengths of exposures over full ODB content. The largest number of exposures lie between 30 and 240 seconds.

Scores for top ranked candidates.		
Rank	Group ID	Score: f_{SU}
1	J0129	0.8266
2	J0926	0.6930
3	sbs909	0.6688
4	a0535	0.6512
5	BDp25_727_sz	0.6389
6	blazars-opt-0420	0.6264
7	blazars-opt-cta26	0.5651
8	2487G000t000	0.5603
9	2428I000t000	0.5248
10	Test	0.4612
11	2392D000t000	0.4470
12	2428J000t000	0.4464
13	2469I000t000	0.4424
14	2476F000t000	0.4214
15	2494J000t000	0.4209
16	2456E000t000	0.4141
17	2505I000t000	0.4052
18	2447A000t000	0.3987

Table 4.4: Scores of top ranked candidate groups for single scheduler sweep at 20:37UT. The group selected on this sweep is *J0129*

displays the typical *staircase descent* between 19:30UT and 21:30UT as the initial population of candidates is used up before the next set become enabled. As can be seen the winning score jumps around quite noticably, this is due to higher scoring groups becoming enabled. The flat sections on the plot represent periods when no scheduling is taking place, either because a group is executing or because operations are suspended for some reason.

Selecting one of the lower ranked groups and for no particular reason chosing the 15^{th} ranked group *2494J000t000*, Fig. 4.30 also shows its score trend relative to the winning group on each subsequent schedule sweep. Its score rises quickly over the next few hours reaching a peak of 0.65 when it is scheduled at 00:06UT.

As can be seen *2494J000t000* missed being selected narrowly on the 2 sweeps previous to its actual selection. Between 21:30UT and 23:00UT a single group is executed, all things being equal our candidate group with its rising score should be selected at 2300UT as its score has ramped up while the general trend of the other candidates is downward. However a set of newly enabled groups have become enabled in the meantime, delaying its selection until the general trend has decayed to the level of *2494J000t000* at 00:06UT

There is thus from the point of view of any particular group an element of chance on when it might actually be selected (if at all) due to the particular distribution of information within the ODB.

4.6 Summary and conclusions

In this section I have investigated the environment in which the scheduler operates in order to determine the character and range of variation of these parameters, how they might affect scheduling and the possibility of making predictions of how these might develop in time. In the case of technical downtime, it was found that these disruptive events occur irregularly and with little chance of prediction.

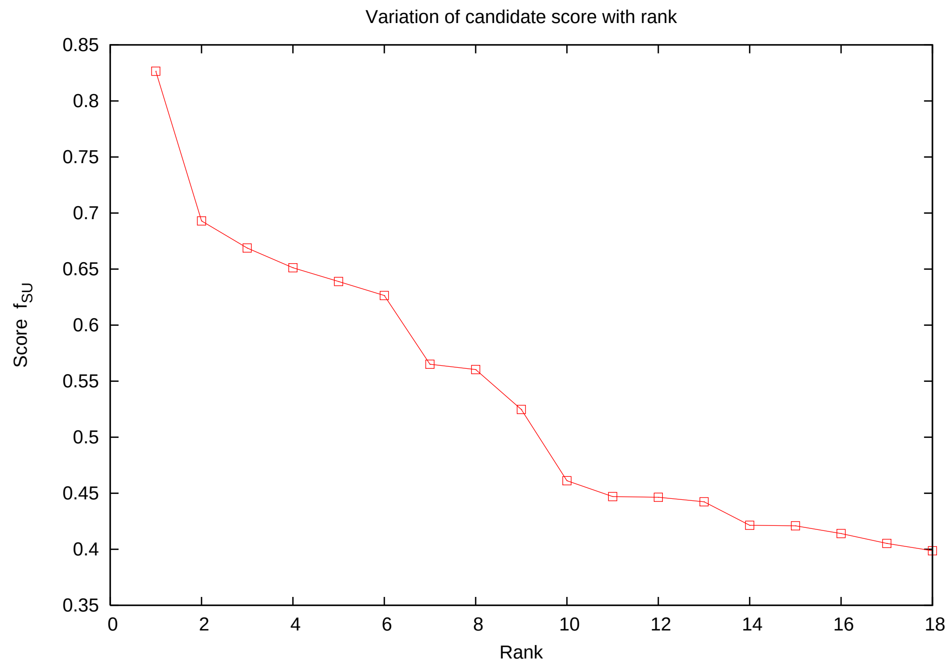


Figure 4.29: Variation of score f_{SU} with rank for candidate groups.

Bad weather which is also disruptive, in that it leads to periods when no scheduling takes place and where schedules already running are interrupted, is dominated by high humidity events accounting to 90% of such periods. A simple model was shown to be capable of predicting the length of a run of good or bad weather for up to 30 hours ahead with $> 80\%$ accuracy.

R-band seeing data, extracted from images taken by the LT's main science camera, was shown to be generally better in summer (average 1.0“) than winter (average 1.5“). Variation of seeing during the course of the night affects scheduling by restricting the set of available groups with reference to their specified observing constraints. Work by various investigators on the prediction of seeing suggests this is a difficult problem due to the nature of the mechanism involved (micro-turbulence in the atmosphere).

The content of the Phase 2 ODB affects scheduling by virtue of the complex interaction between the competing groups of observations and due to the large number of parameters which characterize the pool of observations. An example of such interaction was shown for a particular night which demonstrated the *apparent* randomness of this interaction.

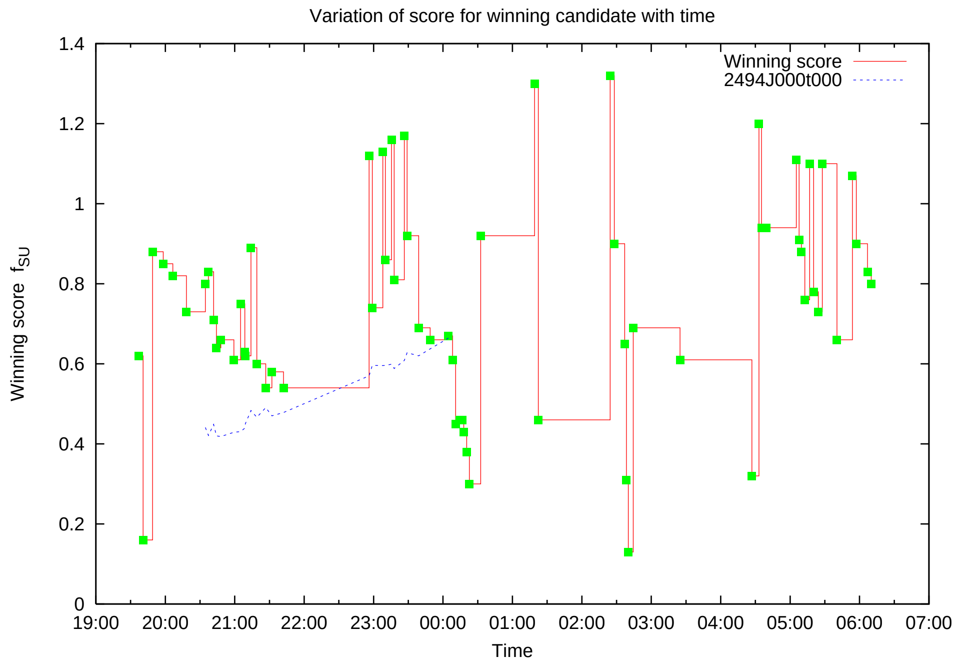


Figure 4.30: Variation of score f_{SU} for group *2494J000t000* relative to winning group on each schedule sweep. This group is ranked 15^{th} on the 20:37UT sweep. Its score rises quickly until 00:06UT when it is selected. Note that the winning score jumps around considerably as higher priority groups become enabled at various times.

5 Architecture

5.1 Introduction

A principal aim of this project was to design a component architecture for building scheduler implementations along with a simulation framework in which to test and measure the performance of these schedulers. As a preliminary step, prior to the start of this project, a simple despatch scheduler Fraser (2004) was built for the LT to allow robotic science operations to start in 2004. The work on this scheduler and subsequent study of its operation provided insight into the range of components that would be necessary to design a scheduler component architecture (SCA).

Some of the objectives in the design of the SCA (Fraser and Steele, 2008) are:-

- Hide the underlying database implementation from the scheduler.
- Provide a summarized version of the database content as the scheduler does not require such detail.
- Provide a range of standard, extendable tool interfaces.
- Hide the scheduler implementation from executor.
- Ability to make predictions about future conditions.
- Time synchronization between scheduler and simulation framework.

It was a principle requirement that all components should be easily interchangable to suit the experiment. Most importantly, it should be feasible to plug an operational scheduler into a simulation environment with very little effort and with **no** modification to the scheduler itself, it should in effect be unaware of whether it is scheduling real observations or not. The design would promote the use of Object Oriented interfaces to facilitate this plugability.

In the forthcoming descriptions the following symbols are used:-

- g A composite group (the group currently under consideration).
- t The current (real or simulation) time.
- a The set of account synopses apertaining to the current group.
- h Synopsis of the execution history of the current group.
- e The current (at time t) environmental conditions.

The SCA is described in Fig. 5.1 and is seen to be built up from a number of distinct layers. Each of these are described in detail in the following sections.

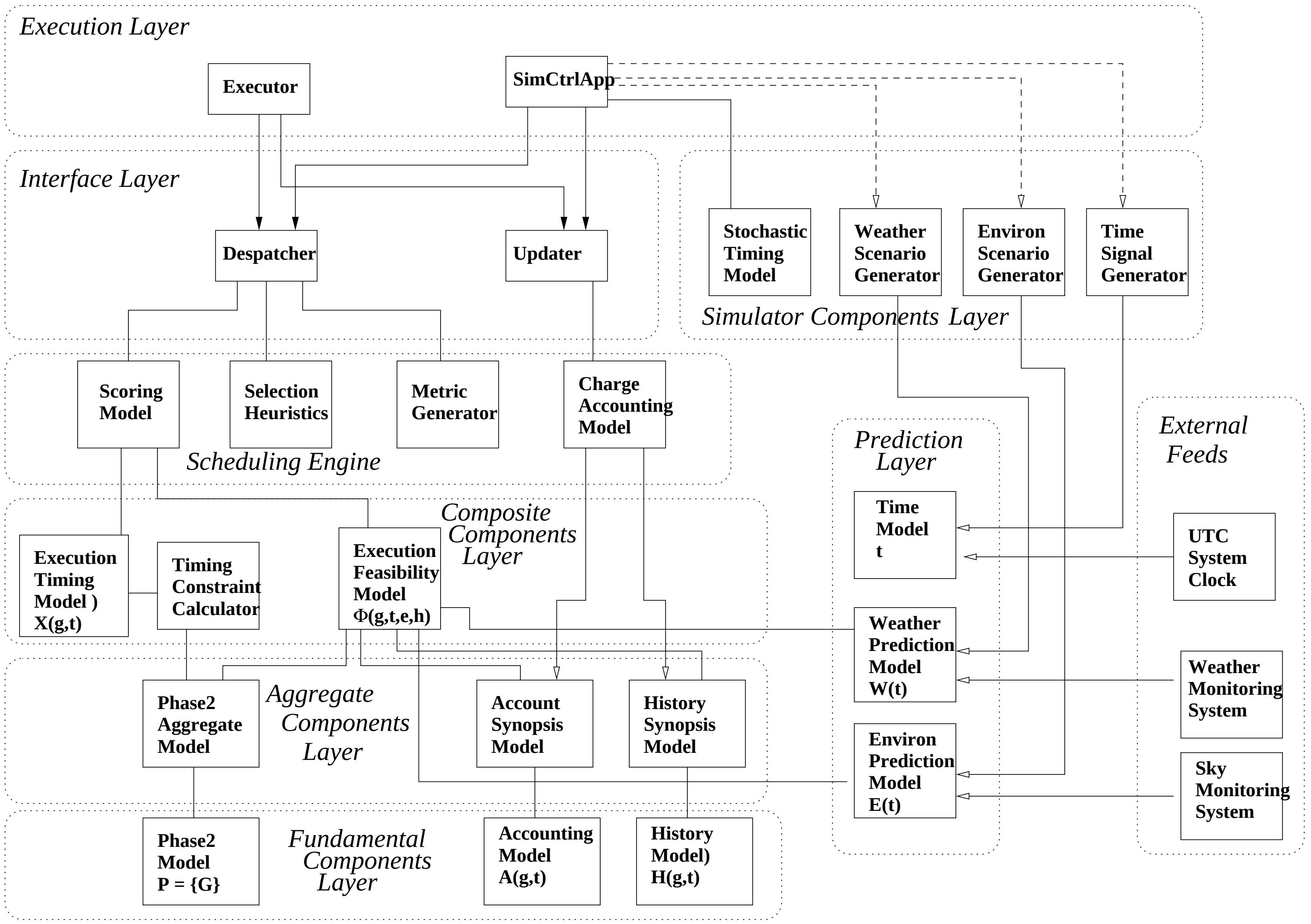


Figure 5.1: The Scheduler Component Architecture (SCA). This is split into a number of layers described in the text.

5.2 Fundamental Components Layer (FCL)

This layer contains models which provide access to the Phase II, Accounting and Group Execution history stored in the ODB. These models are used by the higher layers of the scheduling architecture and by the external Phase 2 User Interface services to query and update the database.

5.2.1 Phase2 model

This is the information entered by the observers or on their behalf by automated agents which describes the content and constraints of the observing programs - i.e what to do and when. The information can be broken into the following categories:-

- Observation specifications contain details of the sequence of operations required to perform the observations; target selection, acquisition and tracking information, instrument selection, calibration and configuration, type, number and length of exposures, mosaicing offset patterns.
- Timing constraints which determine when, how frequently and how many times to perform groups of observations.
- Observing constraints impose limitations on the conditions under which observations may be taken.

5.2.2 History model

The history model (H) represents the record of execution histories of groups from the Phase 2 model. For flexibly scheduled single execution groups this is just the date/time it was executed. For repeating groups it represents the history of all of the times the group has been attempted (whether successfully or not) along with execution statistics.

5.2.3 Accounting model

The allocation and use of resources by groups, proposals and TAGs is provided through the accounting model (A). When a proposal is created and on subsequent semesters if still active, the accounts for the proposal (and sponsoring TAG) are allocated new resources. When groups are executed the balance of relevant accounts is appropriately reduced. When observations are made but found to be sub-standard, the relevant account balance may be adjusted as *payback*. All of these transactions are recorded in the ODB via the accounting model. The model also allows the scheduler or User Interface to trace the history of transactions via an audit trail.

5.3 Aggregate Components Layer (ACL)

The scheduling engine does not use the FCL models directly. Partly because much of the information contained in those models is more detailed or fine-grained than it requires to make scheduling decisions and more importantly for the sake of efficiency, the ACL provides synposes of the FCL models. When running in a simulation environment, rather than generating the basic FCL models, the simulation controller usually generates the models in this layer directly.

5.3.1 Phase 2 Composite Model

The Phase 2 Composite Model (P2C) provides a *group-centric* view of the Phase 2 ODB content. In addition to the sequencing, timing and observing constraints the *composite* groups provided by this model contain details of the owning proposal, TAG, program and PI.

5.3.2 Account Synopsis Model

This model provides a single point *account synopsis* for each proposal containing details of all of the proposal's accounts for each valid semester. The synopses contain balance information but do not provide a detailed audit trail facility.

5.3.3 History Synopsis Model

Details of the latest successful execution and number of executions up to a given date are provided by this model.

5.4 Computational Tools Layer (CTL)

The CTL provides models and tools for processing information derived from the ACL. These are the scheduler's models of the function of the executor, whether RCS or simulation controller. Computing models such as the *execution timing model* $X(g, t)$ which provides details of the time required to execute groups by modelling the function of the execution system and the *feasibility model* $\Phi(g, t, e, a, h)$ which determines the potential observability of groups under given environmental conditions. The astrometric tools can also be considered part of this layer.

5.4.1 Execution timing model

Provides details of resource consumption of groups of observations, answering questions like:- *how long will it take to complete group x?* Information relating to the telescope, instruments and robotic system are combined to make these estimates based on the primitive operations described by the group's observation sequence. Some of these components can be characterized well, others provide a source of uncertainty. In the standard model used in the deployed system, Table. 5.1 describes which events in an observing sequence are used to make the estimation:-

Factors involved in calculation of execution time.	
Time factor	**When required**
Slew rate in each axis	Target changes, position offsets, rotator mode or angle changes, rotator cardinal pointing solution changes.
Instrument filter defocus time	Instrument or filter changes.
Fold mirror move/deploy	Instrument changes, some calibrations.
Instrument configuration	Movement of filter wheel, grating or other internal mechanism.
Instrument calibration	Lamp-flats, darks, biasing, arcs.
Fine-tuning	Acquisition by an instrument onto spectrograph slit
Autoguider acquisition	When switching autoguider on.
Aperture offsets	Instrument changes.
Readout time	Exposures, may depend on binning and windowing
Data write-to-disc time	Exposures, may depend on binning and windowing.

Table 5.1: Factors involved in calculation of execution time for groups.

5.4.2 Execution feasibility model

Generally denoted by the symbol $\Phi(g, t, e, a, h)$ this model is used to determine whether a particular group is feasible subject to its constraints and under specified conditions. A number of factors are taken into account in the operationally deployed system:-

- *Seeing* - The actual seeing, corrected for airmass and wavelength to a standard datum (zenith, r-band) is compared to that specified in any seeing constraint.
- *Lunar elevation* - The moon must be set for groups which specify a Lunar Elevation constraint.
- *Lunar distance* - All targets in a group's sequence must be a minimum distance on the sky from the moon.
- *Hour-angle* - All targets in the group must fall within specified HA limits in order to be observed.

- *Solar elevation* - The sun's elevation is compared to the requested level of twilight or astronomical darkness.
- *Airmass* - All targets in a group must remain above the specified airmass for the duration of the group's execution.
- *Extinction* - Extinction is determined early in the night. If the group requires Photometric conditions and the current conditions are poorer it will not be selected.
- *Horizon* - All targets in a group must be visible above the dome horizon, typically 20 - 25 degrees depending on any engineering settings, for the full duration of the group.
- *Zenith* - Targets must not cross the zenith avoidance zone (ZAZ) during an observation. This is a very narrow zone around the zenith imposed due to speed limitation of the cassegrain rotator tracking.
- *Time limit* - The RCS may impose time limits by which groups must have completed such as for performing important calibration observations. Groups will not be selected if they are expected to overrun into such periods.
- *Daytime* - Groups will not be selected if they are expected to overrun into daytime.
- *Allocation* - A group cannot be selected if the containing proposal's total time allocation will become overdrawn.
- *Activation* - If a proposal is outside of its activation period no groups can be selected from it.
- *Axis Limit* - No target in a group may cross outside any temporary axis range which may be imposed for engineering reasons from time to time.

- *Instrument* - If any observation in a group specifies an unavailable instrument, or a configuration which is not currently available for an instrument, or the instrument is impaired, the group cannot be selected.

- *Fixed group* - If a fixed group is due before a candidate group can be expected to complete (with a short buffer time to allow slewing onto target) then that group cannot be selected.

- *Autoguider* - It is possible to specify mandatory, optional or no autoguider use. If the autoguider is reported as unavailable or impaired, groups which require mandatory use of this will not be selected.

5.4.3 Astrometry

The astrometry library provides tools to work out where various types of target (stars, planets, NEOs and the sun and moon) are on the sky. Additionally it provides tools to determine rising, setting and transit times of objects and various transformations between coordinate systems.

5.5 Prediction Components Layer (PCL)

Contains models for predicting sky and weather conditions and a *time model* which provides a synchronizing time signal. In an operational context these models are fed from external sources such as the weather monitoring system (WMS), the sky conditions model and the instrument and telescope monitoring systems. In a simulation environment these components can be setup to perform predictions of whatever degree of accuracy is required by tying their predictions to scenarios generated by the components in the SCL (Sect. 5.6).

5.5.1 Environmental Prediction Model

During both operational and simulated execution the scheduler needs to have available a prediction of the sky conditions. In the case of a despatch scheduler the requirement is simply for the current conditions. In an operational context this information is generated by reduction of the images from the main imaging camera via a real-time pipeline algorithm (ARI, 2011). These are fed into the model and an exponential averaging applied so that recent reductions have higher weight than older reductions. In the case of a look-ahead scheduler the requirement is to determine both the current conditions and how far into the future these conditions will remain stable in order to deduce a suitable sequence horizon. In a simulation context this information can be provided by an Environmental Scenario Generator (Sect. 5.6.1). The prediction can be made to be as accurate or innacurate as the experiment requires. In an operational context, prediction of environmental conditions is somewhat more difficult, a more detailed discussion on this subject is to be found in Sect. 4.4. In the experiments a number of Environmental Prediction Models are used. Fixed environment models E_{FP}, E_{FA}, E_{FX} etc are used in the simulations in Sect. 8. A more advanced model is described in Sect. 10 in the context of measuring the effect of environmental stability on scheduling.

5.5.2 Weather Prediction Model and Disruptor Model

Disruptions are any events which can stop the execution of the observing program. These include:- bad weather, mechanical, electrical and hydraulic faults and software problems leading to node reboots. If the scheduler knew in advance when these events were going to occur it could take them into account in its decision making process. With the exception of weather for which there is at least some limited potential for future prediction, the other types of event are by their nature unpredictable. This is not to say however that some account cannot be taken of these. We can at least obtain long-term averages of the rates and duration of these types of event and feed this information into the scheduler. In the operational context, a despatcher requires only the current weather, though ideally an estimate of the likelihood of the weather remaining good for the length of a group under consideration would be useful. Weather information is supplied by the telescope's Weather Monitoring System (WMS) via various filtering mechanisms supplied by the RCS. In a simulation context weather and other disruptor information is supplied by a Weather or Disruptor Scenario Generator (Sect. 5.6.2). The weather or disruptor prediction can be made to any desired degree of accuracy required by the experiment. In Sect. 11 the effects of such disruptions are studied.

5.5.3 Instrument Synopsis Model

Information on the state of each of the instruments attached to the telescope and thus available for use in the execution of groups is supplied by the Instrument Synopsis Model (ISM). Where a given instrument is either *offline* or *impaired*, a group using the instrument cannot be selected for execution. By coupling an ISM with a generated scenario (Sect. 5.6.5) any evolving instrument availability scenario can be handled. Because this model is a synoptic model rather than a predictive model it basically answers the question *Is instrument x available at current time t?* rather than the question *Will instrument x be available at future time t?* Some care has to be taken particularly in the implementation of look-ahead schedulers to ensure the scheduler only sees the instrument's known state at the simulation time rather than its true state at a future time which would be known to the scenario generator.

5.5.4 Telescope Synopsis Model

Similarly to the ISM, the TSM provides information about the telescope state. In particular the state of the autoguider system is of interest to the scheduler as groups for which use this instrument is mandatory cannot be scheduled if it is *non-operational*. Similar considerations with respect to scenario generation and visibility of information which apply to the ISM also apply to the TSM. A TSM can be coupled to a TSG scenario (Sect. 5.6.6).

5.5.5 Time Model

This model simply supplies the current time to any components. In an operational context this is represented by the system clock. In a simulation context the time signal is coupled to the Time Signal Generator (Sect. 5.6.3).

5.6 Simulation Framework Components Layer (SCL)

Provides *scenario generators* with which to build a simulation environment incoporating time-varying and random effects (weather and sky conditions). A *stochastic timing model* simulates variable execution timing due to uncertainty in mechanical and software processes while a seperate signal generator provides a time signal to the simulation controller.

5.6.1 Environmental Scenario Generator

This allows an environment (sky conditions) scenario to be represented. It can be taken from actual processed sky conditions data or generated using statistical or other means. From the scheduler's point of view, it is not usually concerned with the precise details of how the seeing value is changing from minute to minute, just the general classification of conditions into extinction as photometric/non-photometric and seeing into one of the 3 or 4 pre-defined bands.

5.6.2 Weather Scenario Generator and Disruptor Scenario Generator

This allows a weather scenario to be represented. It can be taken from actual processed weather data or generated from statistical measurements or other means. As with environmental scenario we only need to know if the weather is good or bad not the details though this level of detail may be needed for prediction modelling.

5.6.3 Time Signal Generator

The Time Signal Generator (TSG) is vital to the operation of the simulation framework. It allows the simulation controller to synchronize with the executing scheduler and any real-time operations being performed. More details of this component are to be found in Sect. 6.

5.6.4 Stochastic Timing Model

Although the execution timing model (X) described in Sect. 5.4.1 allows the scheduler to estimate the duration of a group execution, the actual execution duration may vary for a number of reasons including:-

(*i*) A slew can take anything up to 180 secs, (*ii*) Fine-tuning acquisition onto a spectrograph fibre can take up to 60 secs, (*iii*) autoguider lock can take up to 60 secs, (*iv*) science fold movement between instrument ports can take up to 40 secs, (*v*) imager configuration time depends on the start and end filter positions on the wheel and direction of travel, (*vi*) exposure readout times can vary to a small extent but where a large number of short exposures is performed this can be significant.

The Stochastic Timing Model allows this variation to be modelled in a simulation context. This is the size of the time-step for the simulated completion of a group. The distribution of times around the nominal value calculated by the execution timing model depends on the specific model used. Fig. 5.2 shows the results of 2 methods of implementing a stochastic timing model for a sequence containing 3 slews and 3 instrument configurations. The flat plot is the result of adding a random variation to the total calculated execution time from an implementation of the Execution Timing Model. The curved plot shows the results of modelling each of the variable processes then adding the results.

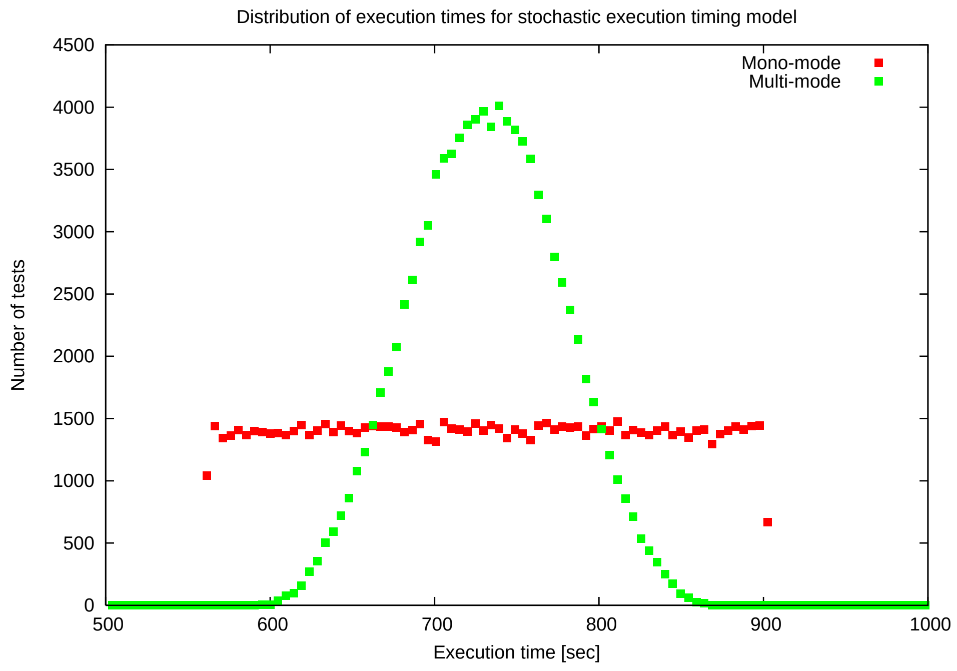


Figure 5.2: Distribution of execution times for stochastic execution model. *Mono* plot uses a single calculation to total up contribution of all events, *multi* plot works out a seperate random effect for each contributing event. In most of the experiments the mono version is used for simplicity.

5.6.5 Instrument Scenario Generator

Allows the time evolving state of the instruments to be modelled. In general this is just setup so that all instruments are always online and available. If there were an experimental requirement to disable one or more instruments for a period such a feature would be implemented here.

5.6.6 Telescope Scenario Generator

Allows the time-evolving state of the telescope systems to be modelled. The most important component being the autoguider status. When this is offline or impaired, any groups with mandatory autoguider use become infeasible.

5.7 Scheduling Engine Component Layer (SECL)

The components in this layer largely determine how a scheduler functions. Implementations of the heuristics and logic of the scheduler algorithms are specified here. In addition to basic scoring and selection mechanisms, other scheduler-specific algorithms would be included here. This layer is in effect what would be described as *the scheduler*.

5.7.1 Scoring Model

Calculation of the score ($S(g, t, e, a, h)$) for a group at a given time and with the specified execution history h, account synopsis a under environmental conditions e from a set of metrics is provided by an implementation of this model . A number of schedule quality metrics were defined in Sect. 3. The derivative versions of these SQMs are frequently used as scoring metrics. Typically for a Q metric such as Q_{OA}, the equivalent scoring metric is denoted $f_{OA} = \dot{Q}_{OA}$.

- *Height metric* f_h simply measures the elevation of the group's target at the time of potential observation. Variations on this metric can include taking the mean or peak elevation over the duration of the group. If a group has multiple targets various averages can be used such as taking mean elevation of each target at some point in the execution. More complex averages could take into account the actual times at which each target will be observed.

- *Airmass metric* f_{air} is similar to f_h except that the airmass is used rather than elevation thus taking into account the nonlinear nature of the relationship between sky-quality and elevation.

- *Transit metric* f_{trans} is an attempt to take into account the unfair advantage provided by the two previous metrics f_h and f_{air}. These force a bias towards targets whose declination is close to the latitude of the observatory. This metric measures the quality of observing at a given time by taking the current target elevation as a fraction of its *best* elevation. This is generally taken as its transit elevation which can be calculated easily. For groups which do not transit during the feasibility window and in particular for groups with very short feasibility windows this may still be somewhat unfair as the transit elevation may not be remotely achievable in that window.

- *Optimal elevation* f_{oh} and optimal airmass f_{oa} metrics attempt to redress this problem by taking the ratio of current elevation to *best possible* elevation (or airmass) in the group's feasibility window.

- *Slew metric* f_{slew} attempts to penalize groups for which a long axis slew (including rotator) from the current telescope position is required. This can prove very difficult to calculate as it requires the scheduler to predict the way the executor will choose rotation angles. The recently introduced cardinal pointing (CP) regime on the LT, to handle the problem of coolant pipes in the wrap, makes this somewhat easier to determine.

- *Sky condition matching metric* f_{see} (also f_{phot} and f_{sky}) is designed to match a group's sky condition requirements to the actual (or predicted) conditions at the time of execution. This is intended to ensure that groups which do not require particularly good conditions do not take an excessive share of good conditions.

- *Lunar condition matching metric* f_{moon} is similar to the above but attempts to prevent groups which can use *bright* time from taking an excessive share of *dark* time.

- *Priority metric* f_p is designed to ensure that groups of higher scientific priority get a better share of time.

- *Resource allocation metric* f_a measures the use of various resources by a group or its containing proposal. This is typically the use of time from the proposal's total semester allowance. It can be used either to help distribute time fairly between proposals or to force completion of proposals which have already been started.

- *Demand metric* f_{td} uses the group's demand for time as an indication of the urgency of performing the group (Eq. 5 in Sect. 3.2.2). The idea behind this is to select the groups which have the most critical requirment for a given time period.

- *Urgency metric* f_{rn} is similar to the demand metric but measures the number of additional chances (in terms of alternative remaining nights on which the group *could* be observed if not tonight.

- *Yield tracking metric* f_{yt} tracks the deficit in data product yield. This is basically the difference between the amount of data that might have been expected if all of a group's observations had been performed at each occasion when they could have been and what has actually been achieved . In order to calculate this metric, both past and future yield estimates and yield to-date are required for each active group. This is an expensive calculation as the set of feasible and available windows have to be calculated over possibly an extended period (e.g. a whole semester).

An actual scoring metric is typically composed of a weighted sum of some of the above metrics of the form:-

$$f(g,t,e,a,h) = w_0 f_0(g,t,e,a,h) + w_1 f_1(g,t,e,a,h) + \cdots + w_n f_n(g,t,e,a,h) \quad (16)$$

where $f(g,t,e,a,h)$ is the score for group g at time t under environment conditions e, account balance a and with execution history h. The weights w_i represent the emphasis that is placed on the various metrics based on management priorities.

5.7.2 Selection Model

Given a set of weighted scores or a set of raw metrics, the Selection Model (ζ) chooses which group or sequence of groups to execute next or over the next horizon. In the experiments (Sect. 9 and others) several selection models are used. In the *Best* selection model, the highest scoring group is always selected. In the various *biased* selection models a degree of noise is added so that sometimes the highest scoring group is not selected. In certain experiments, a self-explanatory *random* selection model is employed.

5.7.3 Metric Generator

This component works out the set of metrics appropriate to a group at a given time under specified conditions. The set of metrics can then be used to calculate a score (ScoringModel) or inspected to apply some other type of criterion to determine a winner. An example is to pick the group for which some metric m_1 is highest, then if tied, select the group for which m_2 is highest.

5.7.4 Charge Accounting Model

When observers prepare their proposals to go before the TAG for time and resource allocation the cost or charging model (C) is used to calculate the amount of time the groups *should* take to execute. In the operational context, standard figures for slewing, acquisition, readout and instrument configuration are published on the telescope website[7] for observers to calculate their resource requirements. These values are used to calculate the nominal cost of performing a group. In reality a group may take more or less time than this but this is the resource usage which is charged to the relevant account. This model is exposed through the Phase 2 User Interface to allow observers to determine these costs in advance.

[7]http://telescope.livjm.ac.uk/Info/PropInst/Phase1/

5.8 Interface Components Layer (ICL)

The executive, whether it be an actual robotic control system or a simulation control application, communicates with the scheduler via the components in this layer. The *despatcher* is the executive's entry point for obtaining a schedule and the *updater* allows the executor to supply observation completion statistics back to the fundamental component models. (e.g. to let the history model know that a group was successfully executed).

5.8.1 Despatcher

The despatcher interface is where the execution system makes requests for new groups of observations to perform. In the case of a despatch scheduler BDS (Sect. 6.4.1) this is the signal to make a scheduling sweep. In the case of an operational Look-ahead scheduler the sequence for the current horizon should already have been generated in advance, the despatcher then just pulls the next group off the pre-compiled sequence. In the case of QLAS (Sect. 6.4.2) the signal to generate a sequence is triggered by the first call to the despatcher. Thereafter it returns the next unexecuted group from the sequence till the horizon is exceeded or the list runs out. It then generates a new sequence on the following call. This would be inefficient for an operational system where the executor might have to wait a significant period at the end of each horizon but is quite acceptable in a simulation environment.

5.8.2 Updater

When a group has completed, whether successful or not, the executor must pass information back down to the database to record the state of this execution and to allow the sponsoring proposal and TAG accounts to be debited. The Updater provides this interface. It may directly modify the ODB via FCL components or may update the cached models in the ACL which then propagate these changes via the FCL to the

ODB. When a group fails for some reason, the history model receives information detailing the cause of the error which can be useful for users and operations staff in diagnosing problems.

5.9 Executive Components Layer (ECL)

Represents the external application which requires the services of the scheduler. This will either be the telescope's Robotic Control System, a simulation control application or a user tool.

5.9.1 Executor

This component represents the system which performs the observations. In an operational system, this is the robotic control system of the telescope. Its general cycle of operation involves repeating the following sequence:-

1. Request next group from *Despatcher*.
2. Decompose observing sequence and send commands to telescope and instruments.
3. Send completion information to *Updater*.

5.9.2 Simulation Controller

For each simulation experiment a controller application must be designed. It is responsible for setting up any generated models and scenarios and for performing the observations. Additional details of simulation operations are provided in Sect. 6.2.

5.10 Implementation

The scheduler component architecture (SCA) is written in Java and consists 195 classes in 16 subpackages totalling 25000 lines of code. The SCA is deployed on both a test system for the simulation experiments and on site on the Observatory Control Computer (OCC). The test system is an AMD Sempron™ 2800+ processor with a nominal speed of 1.6GHz and 1GB RAM. On this system a typical schedule sweep takes around 1 minute. The OCC is a rack-mounted Proliant DL380 with 4 Intel Xeon® processors running at 2.8Ghz and with 3GB RAM. On the OCC a schedule sweep takes from 1 to 4 seconds.

Part II

Experiments

6 Experiments

6.1 Introduction

The simulation experiments are performed using schedulers built with the Scheduler Component Architecture (SCA) described in (Sect. 5). In order to setup a specific simulation experiment it is necessary to devise both a scheduler implementation and a simulation experiment control application (SEC) specific to the experiment (or set of experiments). Fig. 6.1 shows the general architecture of a typical simulation experiment. The SEC creates the various data models (Phase2, accounting, history, environment, etc) and resource tools (execution timing calculator, environment prediction, etc) and makes these available to the scheduler via the simulation engine (simulator). The simulator's operating cycle involves making scheduling requests, performing these and notifying the SEC. The SEC is then responsible for collation of any relevant statistics.

The general format of a night's observing is in the form of a series of events. After each group has started executing there is generally nothing of significance capable of occuring from a scheduling point of view until the group completes or fails. To take advantage of this, the simulator is designed to operate by generation of discrete events. It had been anticipated that the time used in the model could simply be set to run at the maximum possible as permitted by the cycle of $schedule- > execute- > update$.

However, because certain component operations within scheduler implementations might have to run as seperate threads (in the real world), there would have to be a form of time synchronization to allow these types of component to work correctly in the speeded up simulation environment. To illustrate by an example:

A specific scheduler implementation might be required to run a sorting operation which takes perhaps 2 seconds of real time (a long time by simulation standards) every 5 minutes. Now 2 seconds of real time might actually correspond to several minutes or hours of simulation time, resulting in the sorting operation going *out of phase* with the rest of the simulation.

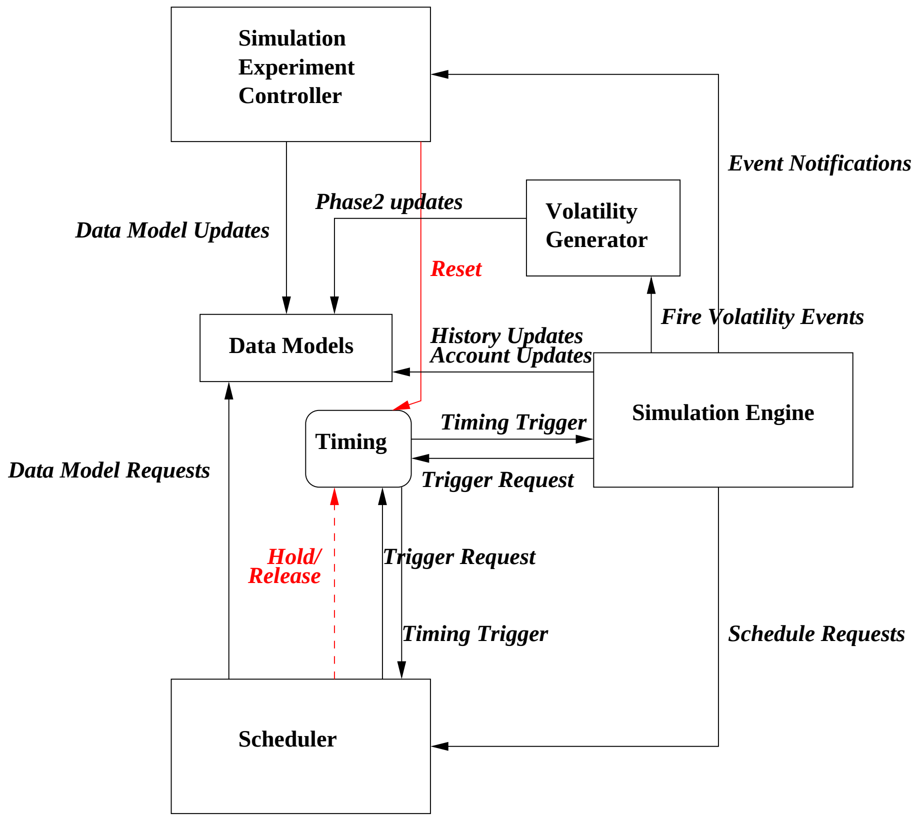


Figure 6.1: The simulation framework architecture. The Simulation Experiment Control application (SEC) is responsible for creating the various data models used by the scheduler. It invokes the simulator to run between specified times and receives feedback of significant scheduling events. The simulator makes scheduling requests of the scheduler and provides updates to the data models when the observations are deemed to have completed. Synchronization between components in speeded up simulation time is provided by the Time Signal Generator (TSG).

This was solved by creating a Timing Signal Generator (TSG). This component permits several operations. In the main, an external client may request an asynchronous time-trigger at some future (simulation) time. The TSG keeps track of trigger requests and sends these signals out asynchronously to the requestors, in order, at the appropriate (simulation) times. Additionally, any client (thread) which requires to perform a *significant* real-time operation may place a hold on the TSG to prevent any triggers being emitted. After completion of the real-time operation the client releases the hold and any queued triggers can be sent out. Ultimately, provided there are no extremely time-hungry real-time operations, the simulation clock advances significantly faster than real-time.

6.2 Simulator Operation

The operation of the simulator cycle is described in Fig. 6.2. The simulator is controlled by an SEC and on initiation runs for a period specified by the SEC. With reference to the figure, a cycle of $schedule- > execute- > update$ is performed. Initially the simulation time t is set to the start time specified by the SEC. A series of tests are then performed to determine *what to do next* and *how long* (denoted τ) this operation will take in terms of simulation time.

The first test determines if t is during daytime. If so the simulation will skip forward to the calculated sunset time. If not, the second test determines if t occurs during any scheduled *disruptive* event (bad weather, mechanism failure, engineering time). If so the time will be advanced to the end of this disruption. If not, an observation group is requested from the schedule despatcher. If none are available, the simulation will skip forward by the length of a *background observation* (effectively idle time). If a group is returned from the despatcher, the *expected* execution time τ_{exec} is determined and a further test is performed to determine of any disruptive events are scheduled to start between t and $t + \tau_{exec}$. If not, the simulation will be advanced to $t + \tau_{exec}$, otherwise to the start of the disruption $t + \tau_{sod}$.

Once the advance (τ) has been determined, but before the time is actually incremented, it is guaranteed that nothing can occur in this interval to affect the run of the simulation. At this point the Volatility Generator is called upon to effect any Phase 2 update events scheduled for the period $[t, t + \tau]$. The simulation time is then advanced by the relevant amount. If a group has been selected for execution it will have either succeeded or failed (it may have been aborted by a disruption event). The various models (history, accounting) are then updated and the SEC receives a notification from the simulator of group completion (or failure).

6.3 Sources of uncertainty

The real environment in which operational schedulers have to function provides many sources of uncertainty (Sect. 4.1). The simulation framework is intended to provide a controlled environment for the scheduler to run in but must provide a realistic degree of variation in these environmental variables. These sources of uncertainty affect both the complexity of the scheduling problem (in terms of complexity metrics) and the range of potential rewards possible (in terms of schedule quality metrics). The main stochastic inputs are:-

Stochastic execution model The *execution resource estimation model* (X) provides an estimate of the amount of time a group is expected to take. The various schedulers use this to decide when a group is likely to complete, however in reality a group might take more or less time due to various sources of uncertainty. The *Stochastic execution model* allows the simulation framework to take these effects into account and introduces some variation in the simulated execution times of groups. This is generally implemented by using the prediction from X then adding some gaussian or white noise to simulate the effect of variable length telescope and instrument operations (Fig. 5.2).

Environmental model On an actual night the seeing and atmospheric extinction may vary over the full range and with a variety of timescales thus affecting the set of

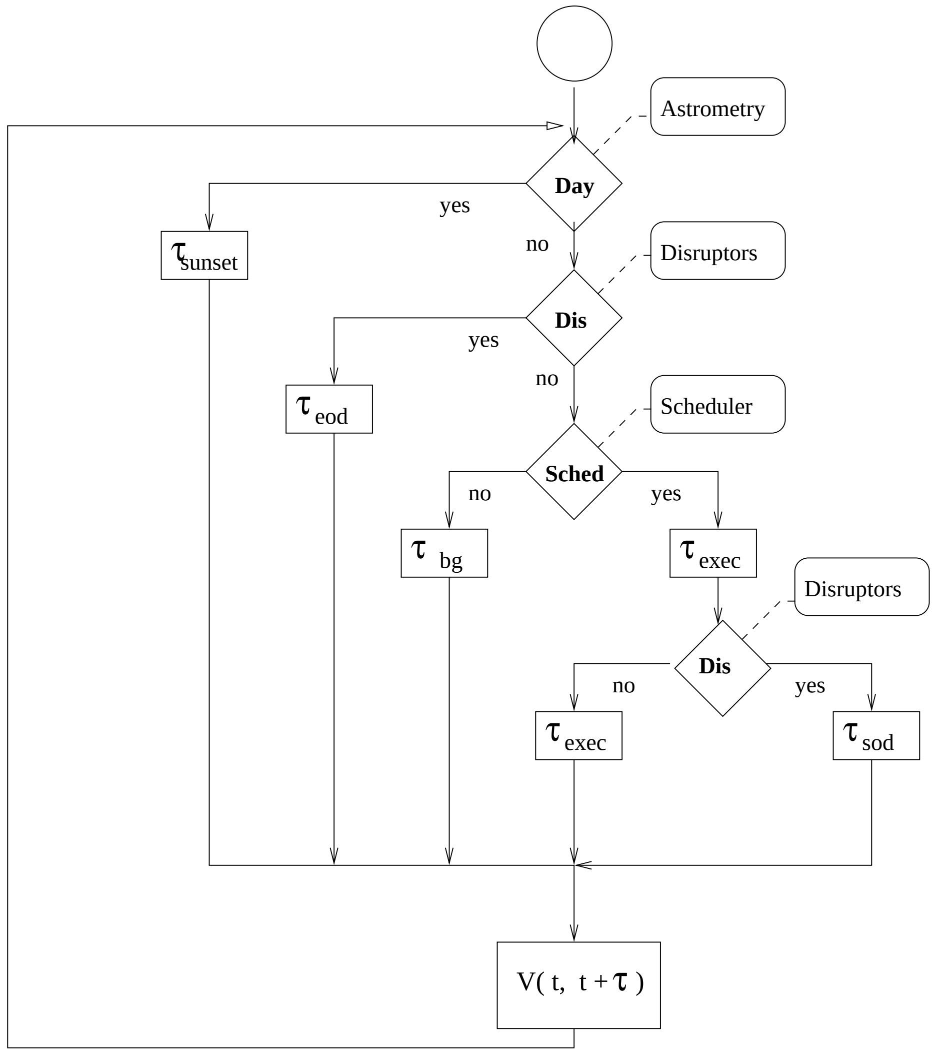


Figure 6.2: Flowchart for simulation control algorithm. The time advance values in each cycle are: τ_{sunset} time remaining until sunset; τ_{eod} time until end of current disruption event; τ_{sod} time until start of next disruption event; τ_{bg} background observing time; τ_{exec} execution time of a group. The decision boxes represent the following tests:- (*i*) $Day(t)$ - is time t daytime?, (*ii*) $Dis(t)$ - is time t inside some disruption period ?, (*iii*) $Sched(t)$ - can the despatcher find anything to do at t?, (*iv*) $Dis(t, t+\tau)$ - are there any disruption events starting during the interval $(t, t + \tau)$. . The volatility generator V is invoked after each cycle to see if any volatile events should occur before the start of the next cycle.

feasible groups available. This natural variation in the sky conditions are taken into account by the *environmental model*.

Phase 2 model The distribution of the generator parameters used to create the Phase 2 information will affect the evolution of a schedule during the night since the Phase 2 model determines the range of groups available over time.

Disruption model Interruptions of varying sizes and lengths due to (unpredictable) events affect the degree of success of schedules, this aspect is simulated using a *disruption model*.

Volatility model Evolution or volatility of the phase 2 database content is simulated using a *volatility model* which allows new groups to be injected into the ODB at (controlled) random intervals thus affecting the contention and other parameters of the Phase 2 model.

6.4 Scheduler implementations

Two different types of scheduler are used in the experiments. These take rather different views of the process.

6.4.1 Despatch Scheduler

Despatch schedulers take a local view of optimization. At any point in time they will try to select the group g which has the highest *differential* score f_g and which best matches current conditions. The algorithm is particularly straightforward:-

1. The *ExecutionFeasibilityModel* is used to generate a candidate list of groups which satisfy their timing constraints and all of their observing constraints under the current environmental conditions.

2. Using the *ScoringModel*, various metrics and differential scores are calculated for each candidate group.

3. The *SelectionModel* is used to analyse the candidate metrics and choose one of these groups to execute.

The baseline implementation used for simulations is called *Basic Despatch Scheduler* (BDS). This implementation is typically configured to use different versions of *SelectionModel* and *ScoringModel*. For most cases the selection model ξ_{BEST} is employed - this selects the highest scoring candidate group. Various biased selection models are tested in Sect. 9.5. The scoring model employed is a weighted sum of a number of standard metrics such that the differential score f_g for a candidate group g at time t under environmental conditions E is given by Eq. (17):-

$$f_g = \sum_i w_i f_i(g, t, E) \tag{17}$$

The *execution timing model* used by BDS has a configurable fixed duration estimate assigned to each type of observing sequence component (see Fig. 1.1 in Sect. 1.1) with the exception of exposures for which the duration is calculated using configurable, instrument-specific overheads. (See Table. 5.1 for more details.)

6.4.2 Look Ahead Scheduler

The Look-Ahead Schedulers (LAS) try to increase the overall reward by optimizing over a period of time (global optimization). They have to make estimates of what is likely to happen in the future - both in terms of external conditions (stability, volatility, disruption) and the likely outcomes of their own actions (how likely are the selected groups to succeed, will they overrun). In the discussion to follow the term sequence refers to the ordered sequence of groups to execute during a given horizon. A typical LAS implementation works as follows:-

1. The *HorizonDeterminant*, using information about environmental stability, ODB volatility and disruption decides how long the sequence horizon (H) should be.

If the horizon is too short we risk losing potential reward. If H is too long, we risk breakages.

2. Using the *SolutionGenerator*, a number of candidate sequences are generated in which each group must satisfy the timing and observing constraints under the predicted conditions at its expected time of execution according to the *ExecutionFeasibilityModel*.

3. The *ScoringModel* and *ExecutionTimingModel* are then used to calculate a potential reward (F_S) for each sequence S. When we work out this score or reward, we are assume that conditions will not change during the duration of the sequence to de-enable any of the chosen groups, that the groups will not overrun and that the sequence will run to completion. This may not however turn out to be the case.

   In order to take into account such possibilities, one technique borrowed from AI is the concept of discounting, specifically the concept of Expected Future Reward (EFR). In calculating potential reward, a discount rate (λ) is selected and applied to the scoring calculations to devalue contributions further into the future (later in the sequence). This derives from the notion that the value of a *certain* reward right now outweighs the value of a potentially higher but uncertain reward at some time in the future. The further into the future and the more uncertain the potential reward, the less we value it. The choice of discounting rate is however the difficult part.

   If an EFR policy is in force (Sect. 5) this would be used, perhaps in conjunction with environment and volatility prediction, to calculate the discount rate (λ). The discounted total reward F_S for the sequence is calculated using Eq. (18):-

$$F_S = \sum_{g \in S} X_g f_g(t_g, e) e^{-\lambda(t_g - t_0)} \tag{18}$$

   where X_g is the expected execution duration of the group g, $f_g(t_g, e)$ is the differential score for g at its expected time of execution t_g under environmental conditions e for a sequence starting at t_0. If EFR is not in use (as is the case for most of the experiments) then $\lambda \equiv 0$.

4. The *SearchMechanism* examines the potential candidate sequences to determine the one with the highest potential reward which is then selected for execution.

5. During execution, if conditions deteriorate, the *BreakagePolicy* decides on the course of action to take. This might involve aborting the sequence , swapping in some alternative groups or just carrying on.

6. If new or modified groups appear during execution, the *SequestrationPolicy* determines whether these will be allowed to enter the sequence . In a real-life scenario it might be that a very important and urgent group might be added at such a time. In particular if the horizon H were very large (eg 4 hours) this new urgent group might well be missed

7. When a sequestration does takes place, the choice of which group(s) to replace is determined by a *RetractionHeuristic*.

Two implementations of LAS are employed in the simulation experiments.

- QLAS (Quantum Look Ahead Scheduler) uses a pre-assigned horizon length (H). Its solution generation proceeds by dividing the available horizon into discrete short time segments or *quanta*, labelled τ_Q). For each time quantum, a random selection is made from all those groups which are feasible at that time. For quanta in which no group is feasible an idle gap is left. Once the horizon is filled, the sequence is scored and the highest scoring sequence after a specified number (N_S) of runs is selected for execution. The operation of QLAS is such that during the execution of a schedule horizon any additional groups added cannot be considered until the horizon is completed - there is no *SequestrationPolicy*.

- ELAS (Enhanced Look Ahead Scheduler) is a modified version of QLAS. The main enhancement is to allow new groups generated during execution to sequester time. The *SequestrationPolicy* uses the mean differential score $\bar{q} =$

F_S/H of the groups included in the sequence to calculate a threshold score value for the horizon. If a volatility event occurs during execution, the new group is checked to determine its feasibility window and differential quality metric q_g. These values along with the time remaining in the sequence are compared to the pre-calculated threshold to decide if the group is allowed to be inserted into the sequence by ejecting one or more previously included groups. The probability P of inclusion of a group with score q_g at time t (measured from the start of the current sequence of length H) and with feasibility window W is given by Eq. (19).

$$P = \begin{cases} g(q_g, \bar{q}, t, H) & : q_g \geq \bar{q} \\ 0 & : q_g < \bar{q} \end{cases} \tag{19}$$

where $g(q, \bar{q}, t, H)$ is a rising function of $q_g - \bar{q}exp(-(1-W-(1+a)t)/H)$ with $g(0) = g_0$, $g(1) = 1$. The effect of this is to prevent groups of low score (relative to the mean score for the sequence) from ejecting any sequenced groups early in the execution. As the execution procedes the threshold is eased so it becomes easier for a new group to jump in. Higher scoring groups are always more likely to be able to jump in than low scoring groups and ensures that groups which are capable of running in the next horizon are less likely to jump in than those which must run during just the current horizon. The swap or retraction rule is effected to allow one or more adjacent groups to be removed with just sufficient time to allow the new group to run but with as little lost time as possible and with the minimum loss of reward as follows:-

- Determine if any single groups of duration $X_1 >= X_g$ exist.
- If so, record the one for which $(X_1/X_g - 1)X_1 f_1$ is smallest.
- Determine if any adjacent pairs of group exist such that $X_1 + X_2 >= X_g$.
- If so, record the pair for which $((X_1 + X_2)/X_g - 1)(X_1 f_1 + X_2 f_2)$ is smallest.

  - Continue to triples and quadruplets etc of groups, recording the lowest scaled reward in each category.
  - Finally, select the single, pair or higher grouping with the lowest loss of total reward.

6.5 Notes on notation used in results

In the following chapters results of simulations are displayed mainly in the form of graphs showing differences in various quality metrics achieved by different schedulers or scheduler variants against some independant variable.

A particular form of graph frequently used is the *candlestick* or *box and whisker* plot. A typical example is shown in Fig. 7.3. In these plots the box area is centred vertically on the average result for the dependant variable for the given independent variable. The upper and lower limits of the box represent the average value $\pm$ one standard deviation of the simulation results. The extension bars above and below the box represent the individual extreme values of the dependant variable found during the simulations.

The statistics derived from the graphed results are found in Appendix B. The majority of the derived tables show the value of the difference in a quality metric between one or more comparison schedulers against the results for some baseline scheduler at a variety of selected values of the independant variable used. These *x* values are typically chosen to display a particular feature of the results, e.g. where a plot suddenly rises from a steady low value followed by a typical data point in the risen region of the graph.

The main statistical values shown in the tables are the values of $\Delta_{90}(C, B)$. This represents the improvement in quality (Q) for the comparison model (C) relative to the baseline model (B). It is in fact the ratio of the mean difference between the chosen quality metric Q_C of the comparison and of the baseline Q_B relative to the baseline measurement. The second figure in each table cell, in parenthesis, is the range of this variable in which 90% of results are expected to lie. e.g. $\Delta_{90}(C, B) = 25(\pm 5)$ indicates that 90% of the results for $(Q_C - Q_B)/Q_B$ lie between 20% and 30% or that C is between 20% and 30% better than B.

7 Man against machine

7.1 Introduction

Selecting appropriate groups to perform is a complex task. There are many trade-offs to be made between the various competing preferences and constraints. When the schedule is heavily loaded, i.e. there are potentially more observations to perform than could be accomodated even under ideal conditions, the job of the scheduler becomes both simpler and more difficult. It is relatively *easy* to find a solution which satisfies the constraints, however finding an optimal solution is rendered more *difficult* due to the number of potential solutions to compare.

A major problem is to determine just what those trade-offs are. We would like to be able to answer questions like:- *how much more important is a high priority, non-urgent group relative to a medium priority, urgent group?* It is because it is difficult to quantify these relative weights that we turn to the human scheduler. Humans are capable of processing this type of complex, interlinked information and making these sort of trade-offs on a daily basis - often with little knowledge they are doing it. If we set up a human to perform the task we might be able to deduce from what they have done and the choices they have made some of the rules they are using (often without realizing) and the relative weightings they employ.

Two sets of trials are performed. Firstly the human scheduler (referred to as HS) is presented with a snapshot of the ODB content for the trial night in a tabular format from which to generate a schedule. Secondly a set of simulations are run in which the relative scoring weights are changed to try and reproduce the results of HS and, *if possible*, to beat them.

7.2 Characterization of problem

On the night of the test (13-14 November 2007) there are some 13.33 hours of night (including twilight time) with additonal details in Table. 7.1. The supplied ODB snapshot consists of 131 groups and a total of 2880.32 minutes (48 hours) of available observing time in the night, an oversubscription factor of 3.6

Case study night characteristics		
Item	Time	Lunar elevation
Sunset	2007-11-13 18:15 UT	23.4°
StartNight	2007-11-13 19:40 UT	11.2°
Moonset	2007-11-13 20:50 UT	0.0°
EndNight	2007-11-14 06:05 UT	-66.0°
Sunrise	2007-11-14 07:35 UT	-47.3°
Item	Duration (hours)	
Length of dark night	10.42	
Length of night inc twilight	13.33	

Table 7.1: Characteristics of the test night for HS1 experiment.

The groups within the snapshot are distributed throughout the range of priority as shown in Table 7.2. Excluding the background groups, which should only run as a last resort and do not contribute to the science program, there are a total of 34.8 hours giving a more realistic oversubscription factor of 2.6.

Figures 7.1 and 7.2 show respectively the contention and demand plots for the night. The predicted contention C_c is very high at around 90 for much of the night. A simulation run with a *typical* scheduler, shown on the same plot shows the actual contention dropping from around 80 to 40 over the course of the night. The total demand C_d remains fairly constant at a level of around 4 to 5 for the night while the urgency-weighted demand C_{ud} rises from 1 to nearly 2 over the course of the night. This latter indicates that selection of just the urgent groups would likely fill most of the available observing time.

7.3 Human scheduling trial (HS1)

A human subject (expert scheduler) is provided with information concerning which groups are potentially feasible on the test night. An example section of the data provided is shown in Table. 7.4. The information, described in Table. 7.3 contains details

ODB snapshot characteristics		
Priority	N_G	$\sum X_g$ (hours)
0	1	2.8
1	55	1190.1
2	26	462.3
3	12	169.5
5	2	39.4
BG	23	790.3
STD	12	226.8
Total	131	2880.3

Table 7.2: Distribution of number of groups N_G and total executable time $\sum X_g$ among the available priority levels. Low priority (1 and 2) make up the bulk of the science groups with relatively few high priority (5) groups. Background (BG) groups which only execute if no others are available and calibration standard (STD) groups are not part of the science program.

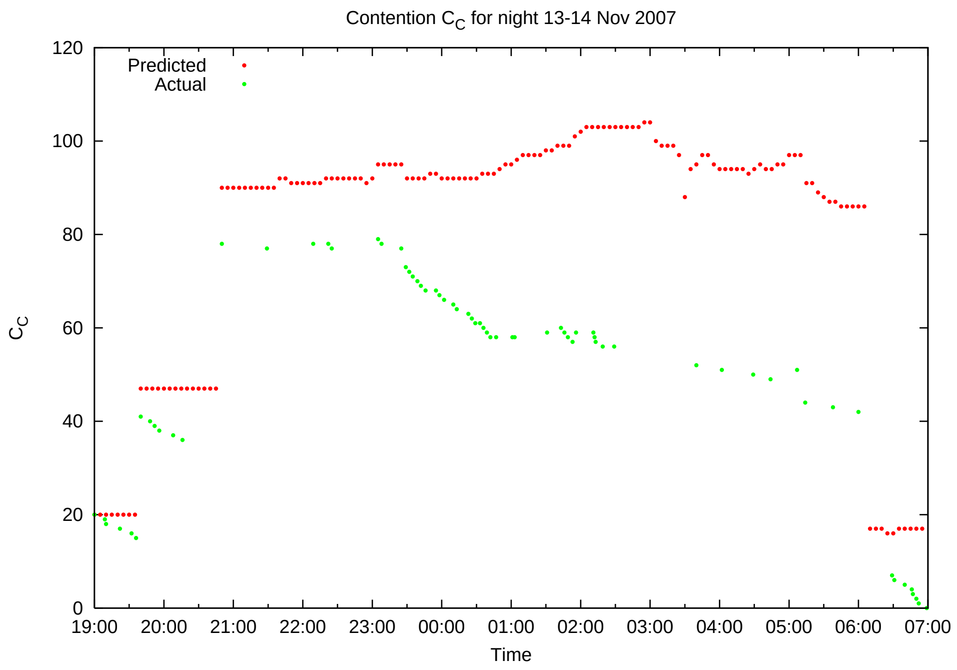


Figure 7.1: Contention for night 13-14 November 2007 for HS1 test. The two plots show the expected contention (C_C) based on ODB content calculated in advance and the actual contention (C_A) derived during a typical scheduling run on the night. C_A displays the typical staircase behaviour as the pool of observations is depleted during the night.

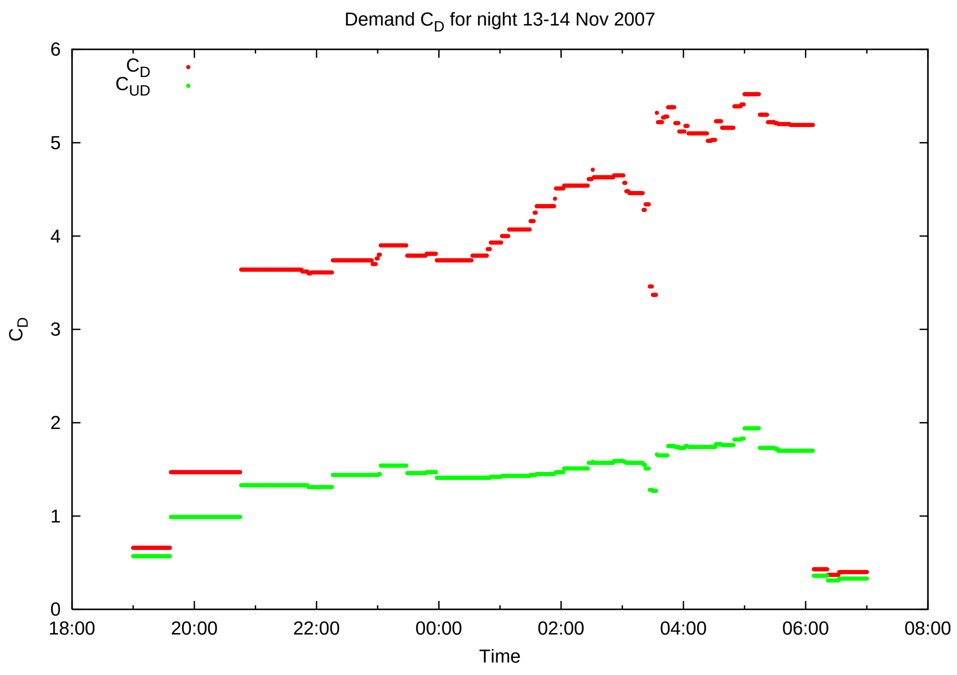


Figure 7.2: Demand for night 13-14 November 2007 for HS1 test. The two plots show the overall demand (C_D)and urgency-weighted demand (C_{UD}). As can be seen there are enough urgent observations to fill the night.

of the target, timing and observing constraints along with a simple pre-calculated feasibility plot for the night for each group. Some metrics are also available. The urgency metric indicates on how many remaining nights the group might be observed. The execution-time metric indicates the expected execution time of the group.

In light of the difficulty for a human to generate the schedule by collating the considerable amount of information available a number of the normal constraints were relaxed. e.g. it is not easy for a human to keep precise track of the accumulating time to the second so execution times are given to the nearest minute. The accuracy of the availability plot is also shown only to around 5-10 minutes so when testing the feasibility (in terms of target elevation) a few degrees of leeway are given. In order to uncomplicate the scenario the assumption is also made that seeing is good for the whole night.

The HS uses the information to devise a schedule for the night by recording the start time and ID of each group to execute in order. Due to the time taken to perform one of

Description of the columns in the table provided to the human scheduler			
Column	**Title**	**Example**	**Description**
1	Numeric ID	G_476	Used by HS to mark up the schedule.
2	Target RA	0:42:49.64	RA (hh:mm:ss) format.
3	Target Dec	41:15:26.50	Declination (dd:mm:ss) format.
4	Group Name	ANGM31	Name of the group.
5	Timing	MONITR 2.5H [2]	Timing constraint class.
6	Moon OC		Observability with moon risen.
7	Seeing OC	POOR	Minimum seeing category.
8	Solar Elev OC		Observability in twilight.
9	Priority	2	TAG assigned priority.
10	Execution Time	33.4M	Expected time to run.
11	Urgency	CRIT	Remaining observable nights.
12	Availability	__9**********_	Display of observability.

Table 7.3: Description of the columns in the table provided to the human scheduler. In the observing constraint (OC) columns, the absence of a label means *no constraint*. The timing constraints include FLEX (flexible - one off), MONITOR (monitor with period and window size (hrs)), INTVL (minimum-interval with interval size(hrs)). If the group *must* be observed tonight (urgency = 1 night) this is shown as either CRIT or ICRIT (minimum-interval groups are never *entirely* critical as they do not also specify a maximum interval). The availability column shows when a group is feasible through the night. Each symbol represents 1/14th part of the night and indicates depending on the symbol what fraction of that period the group can be observed. - (none of period), 9 (< 90% of period), * (< 100%)

.

these HS trials - several hours, only a single trial (HS1) was performed.

7.4 Results of HS1 trial

The analysis proceeds as follows:- For each scheduled (group ID, start-time) pair, the feasibility is worked out for that time. This consists of testing the target elevation against dome limit (20 degs) and the group constraints on moon and sun elevation against actual elevations. If the schedule is feasible, the various quality metrics are then computed.

The HS1 experiment yielded a run of 147 group executions. There were a total of 9 overruns of respectively (1,1,1,1,3,4,5,10,10) minutes, these were considered sufficiently minor to be ignored. A preliminary look at the scoring profile shows the value of $elevation/transit\_height$ to be generally high - suggesting the human scheduler has tried to observe targets as they cross the zenith. There are 2 instances of targets observed below the horizon - one of these at -60 degrees elevation ! - possibly a misreading of a declination? This value was removed from the final results to avoid skewing.

Table 7.4: Sample section of table available to HS. Description of these columns are found in Table. 7.3

G_276	3:24:44.055	50:11:21.63	com17Pmon-071030	MONITR 48H [36]		POOR	ASTR	P= 1	XT= 2.25M	RN= CRIT	3***********9_
G_355	3:13:24.16	18:49:38.40	FS109	MONITR 3H [1]		POOR	ASTR	P= STD	XT= 4.6M	RN= CRIT	_5*********7__
G_369	3:24:7.995	50:9:33.39	17P-mosaic	FLEXBL		AVER	ASTR	P= 3	XT= 13.73M	RN= 21	3***********9_
G_370	3:24:7.995	50:9:33.39	17P-monitor	MONITR 3H [2]		AVER	NAUT	P= 2	XT= 3.63M	RN= CRIT	3***********9_
G_025	4:9:17.00	30:46:33.00	932a000t000	FLEXBL	DARK	POOR		P= 1	XT= 3.27M	RN= 17	__9**********_
G_028	4:30:14.00	35:16:10.00	927e000t000	FLEXBL	DARK	POOR		P= 1	XT= 3.27M	RN= 17	__7**********5
G_093	5:34:32.00	22:0:52.00	927h000t000	FLEXBL	DARK	POOR		P= 1	XT= 3.27M	RN= 16	___2**********
G_382	5:5:30.60	52:49:54.00	zero_G191B2	MONITR 5H [3]		POOR		P= STD	XT= 3.53M	RN= CRIT	__7***********
G_048	6:54:18.125	24:40:7.32	725d000t000	FLEXBL		POOR	ASTR	P= 1	XT= 1.27M	RN= 19	_____*********
G_408	6:54:18.12	24:40:7.38	927c000t000	FLEXBL		POOR	ASTR	P= 2	XT= 1.27M	RN= 19	_____*********
G_002	7:24:18.00	-0:32:17.00	Photom_G1_RU149	INTVAL 3H		POOR	ASTR	P= STD	XT= 9.87M	RN= ICRT	______6*******
G_126	7:8:0.77	22:31:14.50	FocusShort_07	MONITR 168H [167]		POOR	ASTR	P= BGR	XT= 7M	RN= 7	_____7********
G_284	7:23:44.00	20:25:6.00	mho_monitor	INTVAL 22.5H		POOR		P= 1	XT= 71.1M	RN= ICRT	_____4********
G_352	7:24:14.40	-0:33:4.10	FS14	MONITR 3H [1]		POOR	ASTR	P= STD	XT= 5.43M	RN= CRIT	______6*******
G_371	7:36:51.00	65:36:7.00	lens_2403	MONITR 48H [36]		POOR		P= 2	XT= 15.07M	RN= CRIT	____9*********
G_259	8:6:11.10	-27:31:42.00	ESO494-G26	FLEXBL	DARK	POOR		P= 1	XT= 22.17M	RN= 11	________4*****
G_358	8:6:23.70	20:6:31.90	J0806	INTVAL 96H		POOR		P= 3	XT= 27.33M	RN= 17	______6*******
G_472	8:54:48.90	20:6:30.64	0851	MONITR 168H [167]	DARK	AVER		P= 1	XT= 7.8M	RN= 3	_______8******
G_237	9:17:21.90	41:54:38.00	UGC4904	FLEXBL	DARK	POOR		P= 1	XT= 22.17M	RN= 13	______1*******
G_260	9:32:6.20	8:26:31.00	NGC 2906	FLEXBL	DARK	POOR		P= 1	XT= 23.83M	RN= 12	________8*****
G_263	9:45:48.30	-14:22:6.00	NGC2993	FLEXBL	DARK	POOR		P= 1	XT= 22.17M	RN= 10	_________7****
G_266	9:54:28.60	-25:42:12.00	NGC3054	FLEXBL	DARK	POOR		P= 1	XT= 23.83M	RN= 9	__________8***

G_280	9:50:31.97	-2:49:47.20	group07b01c	FLEXBL	DARK	AVER		P= 1	XT= 28.8M	RN= 11	________1*****
G_281	9:47:51.21	-2:24:18.80	group07b01b	FLEXBL	DARK	AVER		P= 1	XT= 28.8M	RN= 11	________2*****
G_373	9:55:33.20	69:3:55.00	lens_M81	MONITR 48H [36]		POOR		P= 2	XT= 15.07M	RN= CRIT	______8*******
G_004	10:50:5.70	-0:1:12.00	Photom_G1_1047+003	INTVAL 3H		POOR	ASTR	P= STD	XT= 9.63M	RN= ICRT	_________2****
G_265	10:52:11.40	32:57:2.00	NGC 3430	FLEXBL	DARK	POOR		P= 1	XT= 23.83M	RN= 11	________3*****
G_282	10:11:42.74	-2:58:16.90	group07b01d	FLEXBL	DARK	AVER		P= 1	XT= 28.8M	RN= 10	_________8****
G_407	10:39:57.45	10:4:0.47	926j000t000	FLEXBL		POOR	ASTR	P= 2	XT= 1.31M	RN= 19	_________8****

Small sample of the table available to human scheduler. The full table has 137 rows. Rows are ordered by the RA of the group's target as can be seen in the final (observability) column.

7.5 Simulation trials

Several simulation trials were performed using a basic despatch scheduler (BDS) and several configurations of a look-ahead scheduler (QLAS). The BDS simulations were performed using different weighted scoring functions. The QLAS simulations were performed using horizons of 1, 2, 4 and 8 hours and using search parameters of 100, 250 and 2500 trials. Each simulation was run 100 times to obtain statistical values for the various Q metrics.

A first set of trials were run using the scoring function $w_{el} f_{el} + (1 - w_{el}) f_{pr}$ with w_{el} ranging from 0.0 to 1.0. With $w_{el} = 0$ the scoring is purely based on priority. At the other extreme ($w_{el} = 1$) the scoring is purely based on target elevation. Plots of the quality metrics Q_{el} and Q_{pr} against w_{el} are shown in Figures. 7.3 and 7.4 with accompanying statistics in Tables B.5 and B.6.

The results of HS1 and BDS simulations with a random selection model are plotted for comparison. From Fig. 7.3 it is clear that random selection is always a poorer option. The HS wins over BDS (29.4% $\pm$4%) when w_{el} is small (priority-dominated selection) and only after $w_{el} > 0.9$ is BDS better (17.5% $\pm$3%) - this suggests that the HS is quite elevation-dominated. Inspection of Fig. 7.4 shows again that random selection is a poor choice. However we also see that HS beats BDS only when w_{el} exceeds 0.7, suggesting that HS has a lower degree of priority bias. From this we may conclude that HS is using a scoring criterion with a large value of w_{el} and a somewhat lower value of w_{pr} though it is not feasible to determine exactly what these are.

A further series of simulations were performed using BDS with fixed values of w_{el} and w_{pr} to gauge the variation in results. The results for these are shown in Table. 4(a). The column labelled **h1** represents HS1. The various columns labelled $\mathbf{B_a}$ represent the use of BDS with scoring based solely on metric f_a. The columns $\mathbf{B_{x,y}}$ represent BDS scored using a combination of f_{pr} and f_{el} in the proportion $(x f_{el} + y f_{pr})/10$.

The scheduler B_{el} achieves the best results for Q_{el} as might be expected. Interestingly it performs slightly less well than HS on Q_{rn} and Q_{td}, and indeed HS performs better on this metric than any BDS implementations including B_{rn}. This suggests there may be an element of *urgency-weighting* in HS. B_{pr} performs best on Q_{pr}, however, surprisingly slightly less well than HS overall on this metric. HS holds its own against all the BDS implementations on the other quality metrics with the exception of Q_{td} where B_{td} performs twice as well as anything else.

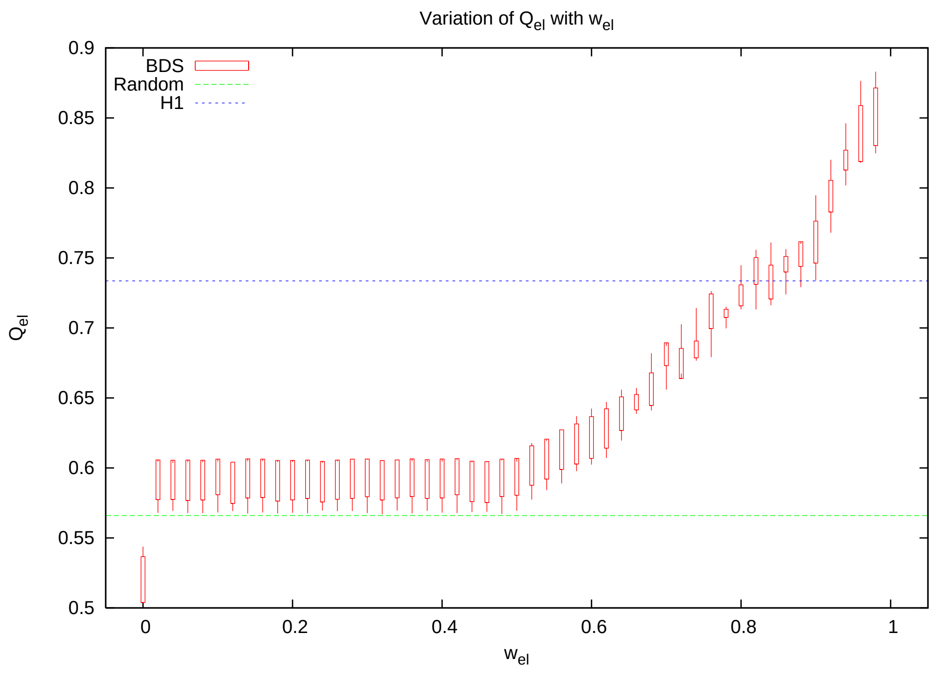


Figure 7.3: Variation of Q_{el} with w_{el} for BDS simulations with scoring funtion $w_{el}f_{el} + (1 - w_{el})f_{pr}$. Plots labelled *H1* and *Random* represent the results for HS1 and for BDS with a random selection model.

7.6 Summary and conclusions

It has been shown that a human scheduler using rules which may not be known to the scheduler himself is capable of generating effective, high-quality schedules as measured by various Q metrics. In some cases the human scheduler was able to outperform an automated (BDS) scheduler on certain metrics. Inspection of the table columns shows the scheduler $B_{8,2}$ comes fairly close to matching the HS column suggesting the relative weightings used are given approximately by $0.8f_{el} + 0.2f_{pr} + \epsilon f_{rn}$ where ϵ is very small ($\ll 0.1$).

(a) Results of BDS simulations and Human scheduler

metric	h_1	B_{el}	B_{pr}	B_{rn}	B_{td}	$B_{2,8}$	$B_{5,5}$	$B_{8,2}$	B_{rnd}
Q_{el}	0.77	0.88	0.52	0.36	0.4	0.59	0.58	0.72	0.56
Q_{pr}	2.34	1.87	2.29	1.73	1.54	2.3	2.3	2.16	1.43
Q_{sm}	0.28	0.3	0.26	0.28	0.32	0.25	0.25	0.25	0.33
Q_{lm}	0.71	0.8	0.68	0.82	0.92	0.64	0.63	0.7	0.86
Q_{rn}	0.67	0.48	0.4	0.58	0.4	0.47	0.47	0.45	0.35
Q_{td}	0.07	0.06	0.05	0.1	0.12	0.06	0.06	0.05	0.08
Q_{xt}	0.99	0.99	0.96	0.99	1.0	0.96	0.97	0.96	1.0

(b) Results of LAS simulations and Human scheduler (n=100)

metric	h_1	**1**	**2**	**4**	**8**
Q_{el}	0.77	0.574	0.569	0.579	0.571
Q_{pr}	2.34	1.848	1.801	1.811	1.812
Q_{sm}	0.71	0.815	0.820	0.847	0.839
Q_{lm}	0.28	0.317	0.321	0.325	0.327
Q_{rn}	0.67	0.362	0.344	0.347	0.358
Q_{xt}	0.99	1.000	0.998	1.012	1.007

(c) Results of LAS simulations and Human scheduler (n=500)

metric	h_1	**1**	**2**	**4**	**8**
Q_{el}	0.77	0.569	0.576	0.584	0.584
Q_{pr}	2.34	1.863	1.818	1.790	1.807
Q_{sm}	0.71	0.816	0.831	0.838	0.843
Q_{lm}	0.28	0.311	0.326	0.329	0.330
Q_{rn}	0.67	0.345	0.357	0.356	0.344
Q_{xt}	0.99	0.994	1.010	1.016	1.011

(d) Results of LAS simulations and Human scheduler (n=2500)

metric	h_1	**1**	**2**	**4**	**8**
Q_{el}	0.77	0.573	0.577	0.58	0.582
Q_{pr}	2.34	1.855	1.826	-	-
Q_{sm}	0.71	0.837	0.838	-	-
Q_{lm}	0.28	0.324	0.327	-	-
Q_{rn}	0.67	0.340	0.342	-	-
Q_{xt}	0.99	1.013	1.011	-	-

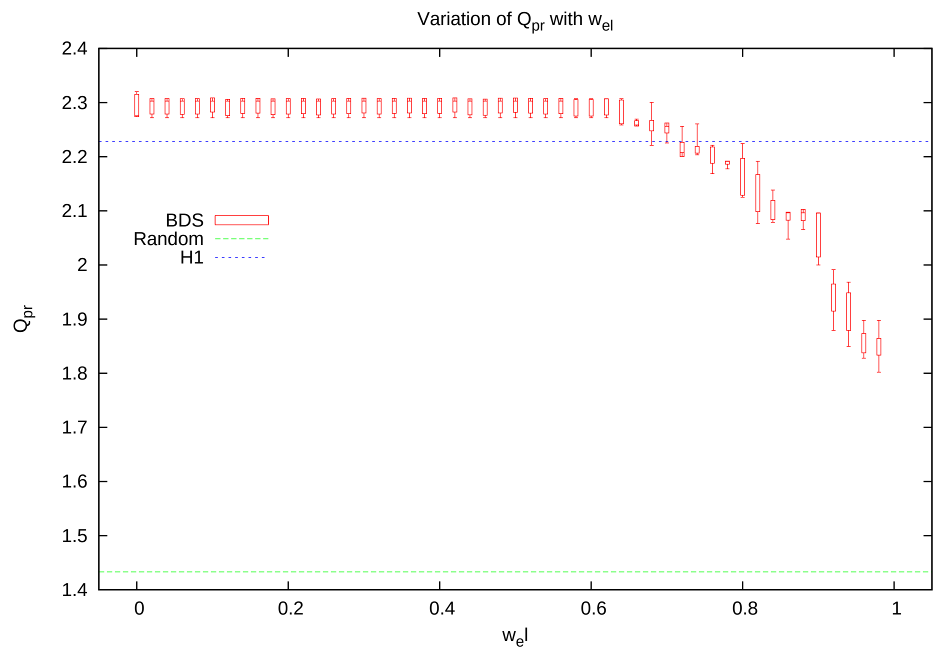


Figure 7.4: Variation of Q_{pr} with w_{el} for BDS simulations with scoring funtion $w_{el}f_{el}+(1-w_{el})f_{pr}$. Plots labelled *H1* and *Random* represent the results for HS1 and for BDS with a random selection model.

8 Investigation of effects of varying scoring weights

8.1 Introduction

This investigation involves running a series of one-night simulations under different environmental assumptions to see the effect of varying the relative values of the scoring heuristic weights. The rational behind this investigation is to provide some initial insight into the usefulness of the various *Schedule Quality Metrics* (SQM) and to provide a shakedown test of the simulation framework on real data. By selecting 2 seperate but similar nights (in terms of loading) it should be possible to get a preliminary idea on the range of likely values and variation of these metrics.

8.2 Method

Two nights were considered and are detailed in Table. 8.1. On night *#1* (27th-28th September 2007), the moon rises early and is up for the full night. There are 12 hours of night of which 9.5 hours are astronomically dark. Night *#2* (3th-4th November 2007) has 5.5 hours of lunar dark time at the start of the night during the 13.0 hour night of which 10.2 hours are astronomically dark.

A series of simulations were run for fixed environmental scenarios E_{FX} (fixed, excellent seeing) and E_{FP} (fixed, poor seeing) on each of the two nights. The scoring model was chosen to be:-

$$f_g = w_{trans} f_{trans}(g,t,E) + w_{px} f_{px}(g,t,E) \tag{20}$$

The weights of the *transit elevation* function w_{trans} and the *execution priority* function w_{px} were varied for each set of 1000 simulations such that $w_{trans} + w_{px} = 1$. These two functions were chosen as they are generally (operationally) the highest weighted functions and because f_{px} does not vary in time whereas f_{trans} does vary with time for a given group. No other functions were incorporated into the scoring calculation in order to avoid any unnecessary noise.

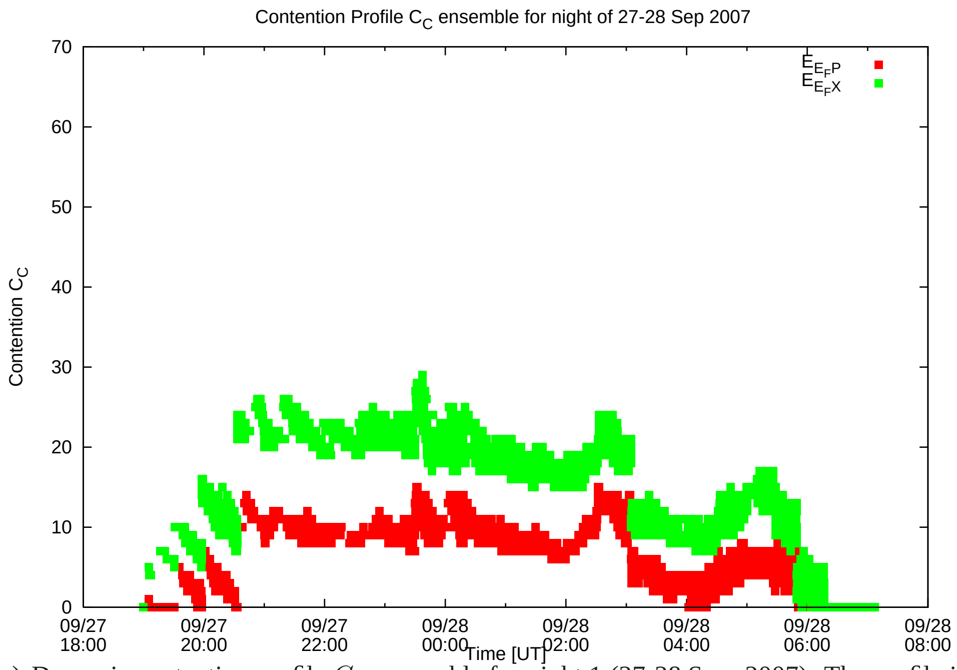
The SQMs used to evaluate the quality of the resulting schedules are Q_{OA} - the optimal airmass metric, Q_{PX} - the execution priority metric, Q_{RN} - urgency metric, Q_{TD} - demand metric, $Q_{PX \cdot OA}$ - priority weighted optimal airmass and Q_{XT} the total execution time as a fraction of available night.

In addition, for each set of simulation parameters an additional set of simulations were performed using a random selection from the available feasible groups.

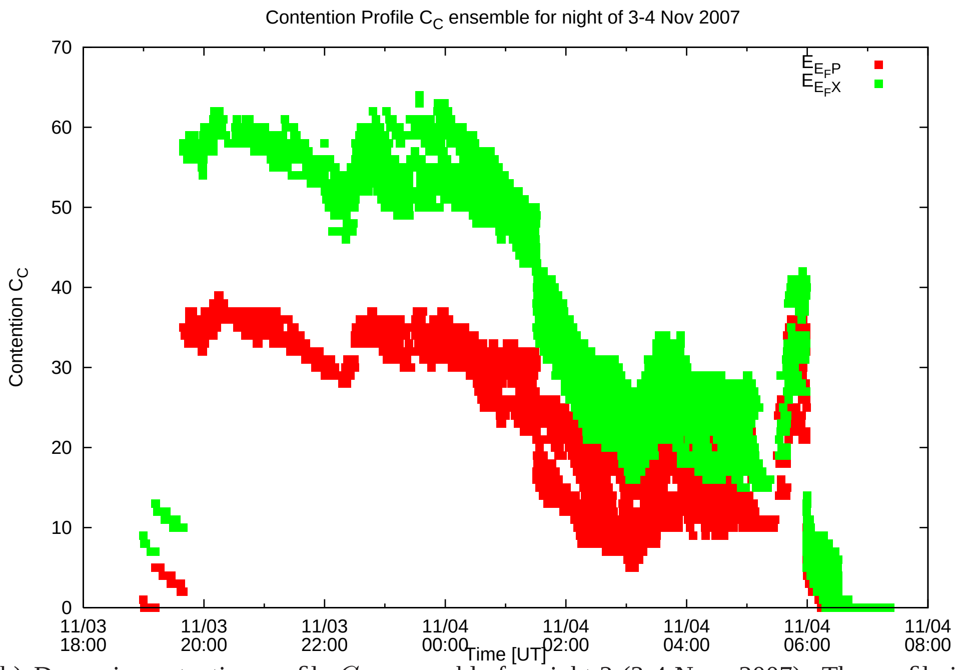
Case study night characteristics		
Item	Night 1	Night 2
Date	27-28 Sep 2007	3-4 Nov 2007
Sunset	18:59 UT	18:22 UT
Sunrise	06:59 UT	07:21 UT
Start Astro night	20:17 UT	19:42 UT
End Astro night	05:41 UT	06:01 UT
Moonset	08:43 UT	14:55 UT
Moonrise	19:23 UT	01:37 UT
Length of night(hh:mm)	12:02	12:59
Length astro night(hh:mm)	9:24	10:19
Length astro moon dark(hh:mm)	0:0	5:56
Lunar dark fraction	0.0%	58.8%

Table 8.1: Case study night characteristics for scoring experiments.

To determine the range of contention statistics for the 2 study nights, dynamic contention data was extracted from the results of a number of simulations. These are presented as ensembles in Figures 8.1(b) and 8.1(a).



(a) Dynamic contention profile C_{dc} ensemble for night 1 (27-28 Sep. 2007). The profile is low for both regimes of environmental condition due to the moon being visible for most of the night.



(b) Dynamic contention profile C_{dc} ensemble for night 2 (3-4 Nov. 2007). The profile is higher early in the night for both environmental regimes but drops to a level similar to that of night 1 after 01:30 when the moon rises.

Figure 8.1: Contention profile ensembles for case study nights 1 and 2.

8.3 Results

The results for the quality measurements for the 2 nights under poor and good seeing are presented in Figures. 8.2(a) through 8.10(b) with supporting statistics in Tables. B.7 through B.18. Figures 8.2(a) through 8.7(a) representing results for night#2 are in pairs, the first figure represents the metric under environment model E_{FP}, the second under E_{FX}. The next set of figures (8.8(a) through 8.10(b) show both sets of environmental condition on the same graph for Night#1.

Figs.8.2(a) and (b) and Tables. B.7 and B.8 show that the Q_{PX} metric is unaffected by the choice of scoring metric. Only when the scoring metric is height dominated ($w_{trans} \rightarrow 1.0$) is there a very slight decrease in the overall priority measure of the scheduler from 37% ± 13 better than random at $w_{trans} = 0.5$ to 29% ± 13 at $w_{trans} = 0.9$. The implication being that the ODB population contains a fairly wide range of priorities and that these are sampled evenly.

Figs. 8.3(a) and (b) and Tables. B.9 and B.10 show the effect on the airmass metric Q_{OA}. We see that as soon as the scoring becomes dominated by airmass ($w_{trans} > 0.5$), the overall airmass metric quickly rises as might be expected from 2.3% ± 3 better than random at $w_{trans} = 0.5$ to 10% ± 3 at $w_{trans} = 0.9$.

Fig. 8.4(a) and (b) and Tables. B.11 and B.12 show a quite different effect for the subset of the population which are flexibly timed groups. Here we see very little change in the value of Q_{OA} even when the scoring is airmass dominated. Only when the scoring has no airmass component ($w_{trans} = 0.0$) do we see this metric drop below its typical value (-3% ± 4 relative to random baseline).

Figs. 8.5(a) and (b) and Tables. B.13 and B.14 show the effect on Q_{PX} of scoring metric for the flexibly timed population. Here we see a much reduced overall priority score relative to the non-flexibly timed population with a distinct and surprising increase as ($w_{trans} \rightarrow 1.0$). At low values of w_{trans}, the metric is reduced by upto 46% ± 19 relative to random selection whilst at $w_{trans} = 0.9$ it is slightly better than random

Results for Random selection model ζ_{random}						
Metric	Q_{OA}	Q_{PX}	Q_{TD}	Q_X	Q_{RN}	$Q_{PX \cdot OA}$
Average	0.835	1.332	2.957	0.907	23.976	1.086
Minimum	0.775	0.921	1.417	0.881	15.331	0.765
Maximum	0.887	1.698	5.000	0.932	33.225	1.417
SDev	0.0225	0.1357	0.8021	0.0081	3.6253	0.1258

Table 8.2: Results for Night 1 under E_{FP}

Results for Random selection model ζ_{random}						
Metric	Q_{OA}	Q_{PX}	Q_{TD}	Q_X	Q_{RN}	$Q_{PX \cdot OA}$
Average	0.817	1.217	2.851	0.929	21.339	0.999
Minimum	0.771	0.817	1.096	0.908	10.410	0.643
Maximum	0.875	1.510	4.842	0.956	30.924	1.346
SDev	0.023	0.1268	0.7203	0.0079	3.5596	0.1179

Table 8.3: Results for Night 1 under E_{FX}

Results for Random selection model ζ_{random}						
Metric	Q_{OA}	Q_{PX}	Q_{TD}	Q_X	Q_{RN}	$Q_{PX \cdot OA}$
Average	0.826	0.966	3.580	0.860	23.605	0.797
Minimum	0.782	0.644	1.916	0.834	16.081	0.548
Maximum	0.866	1.233	5.236	0.875	30.205	1.065
SDev	0.0165	0.1174	0.6754	0.0063	2.6545	0.1013

Table 8.4: Results for Night 2 under E_{FP}

Results for Random selection model ζ_{random}						
Metric	Q_{OA}	Q_{PX}	Q_{TD}	Q_X	Q_{RN}	$Q_{PX \cdot OA}$
Average	0.811	0.812	3.930	0.900	28.765	0.669
Minimum	0.745	0.575	2.079	0.885	20.535	0.439
Maximum	0.864	1.118	6.264	0.908	40.172	0.908
SDev	0.0221	0.1123	0.8007	0.0038	3.746	0.1020

Table 8.5: Results for Night 2 under E_{FX}

by 8% ± 18. This suggests that in *this* particular population the flexible, high priority targets are mostly at quite low declination.

Figs. 8.6(a) and (b) and Tables. B.15 and B.16 show the effect on the target-demand quality metric Q_{TD}. We see a very wide range of values for all scoring models with no real trend. The value of Q_{TD} scarcely exceeds the level found with random selection, e.g. at $w_{trans} = 0.55$ the improvement on random selection is 53.6% ± 32.5

Figs. 8.7(a) and 8.7(b) and Tables. B.17 and B.18 show the effect on urgency via Q_{RN}. There is a suggestion that urgency in this population correlates to some extent with airmass - the higher targets are slightly more urgent. This metric always exceeds the result from random selection by a good margin with 59% ± 16 improvement over random at $w_{trans} = 0.1$ decreasing to 33% ± 16 at $w_{trans} = 0.95$.

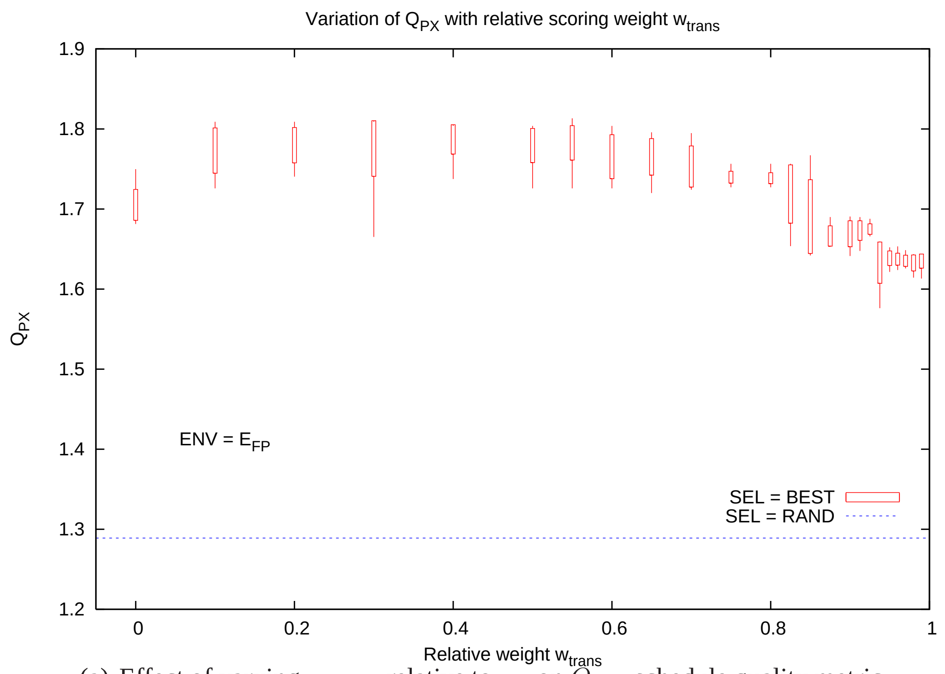
Figs. 8.8(a) and (b) show the effect on Q_{OA} and Q_{PX} respectively for night#1. We see similar results to Figs.8.2(a) and 8.3(a).

Figs. 8.9(a) and (b) show results for night#1 for QRN and Q_{TD}. Unlike night#2 where Q_{TD} showed little variation, on night#1 it is seen to have a negative correlation with airmass. QRN behaves similarly to night#2.

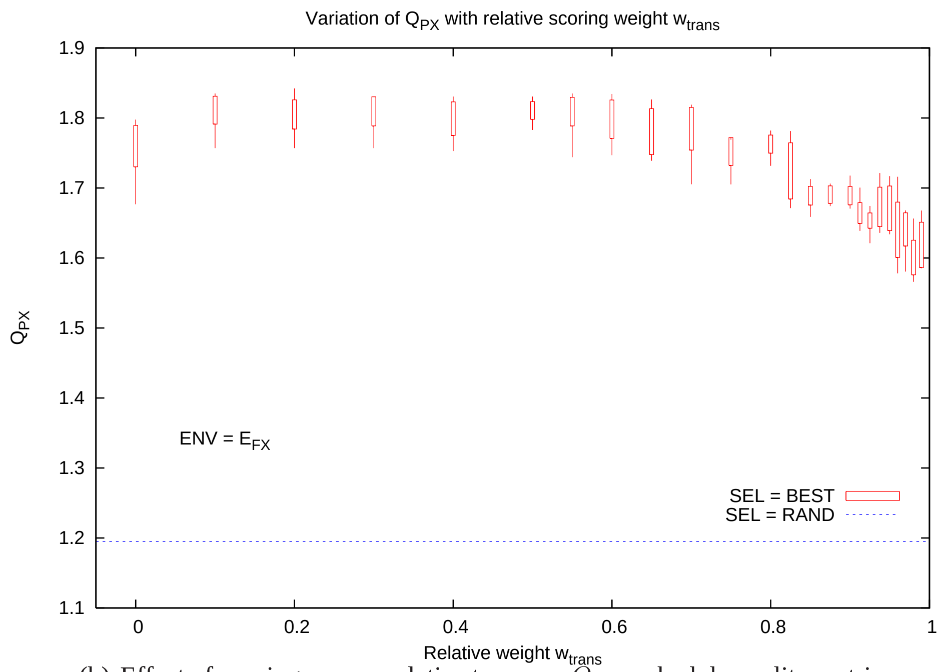
Figs. 8.10(a) and (b) show the effects on Q_{OA} and Q_{PX} for flexible groups. The behaviour is similar to night#2 with Q_{PX} having little variation.

8.4 Summary and conclusions

This series of investigation has shown that the effect of changing the weights applied to the various heuristic metrics used in making scheduling decisions have little effect on each other. When we change any given scoring (f) metric the corresponding quality (Q) metric is affected though not as much as might have been anticipated. There is generally little effect on any other Q metric. The choice of weighting determines the character of the schedule but only the dominant f metric has any real effect while the other metrics appears like noise and suppress the effectiveness of the chosen f metric. We also see the effects of particular distributions of phase 2 information can have an effect on the various metrics, e.g. we can deduce from Fig. 8.5(b) that flexible, high priority targets in this population are mostly at quite low declination. The overall conclusion that can be taken from this is that whatever qualities we want from a schedule must be actively selected for in the scoring and selection models.

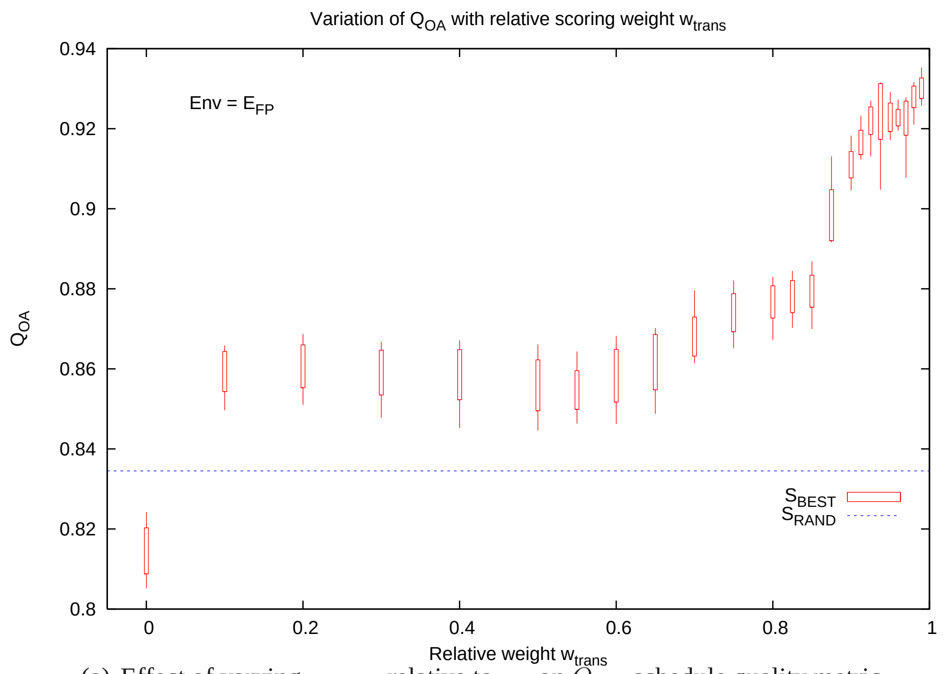


(a) Effect of varying w_{trans} relative to w_p on Q_{PX} schedule quality metric

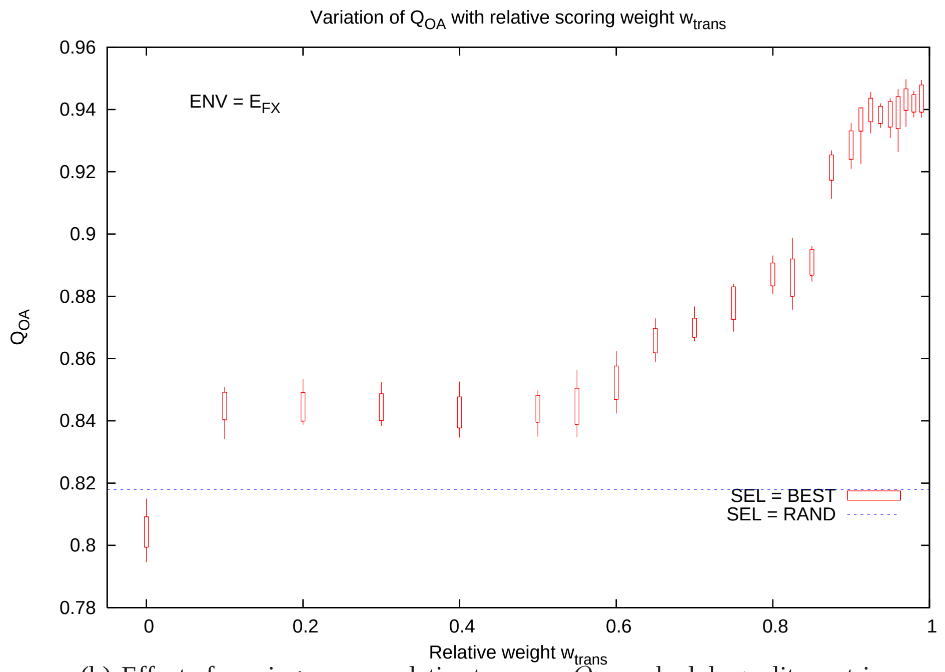


(b) Effect of varying w_{trans} relative to w_p on Q_{PX} schedule quality metric

Figure 8.2: Variation of Q_{PX} with w_{trans} for environment models E_{FP} and E_{FX}. Q_{PX} metric is unaffected by the choice of scoring metric.

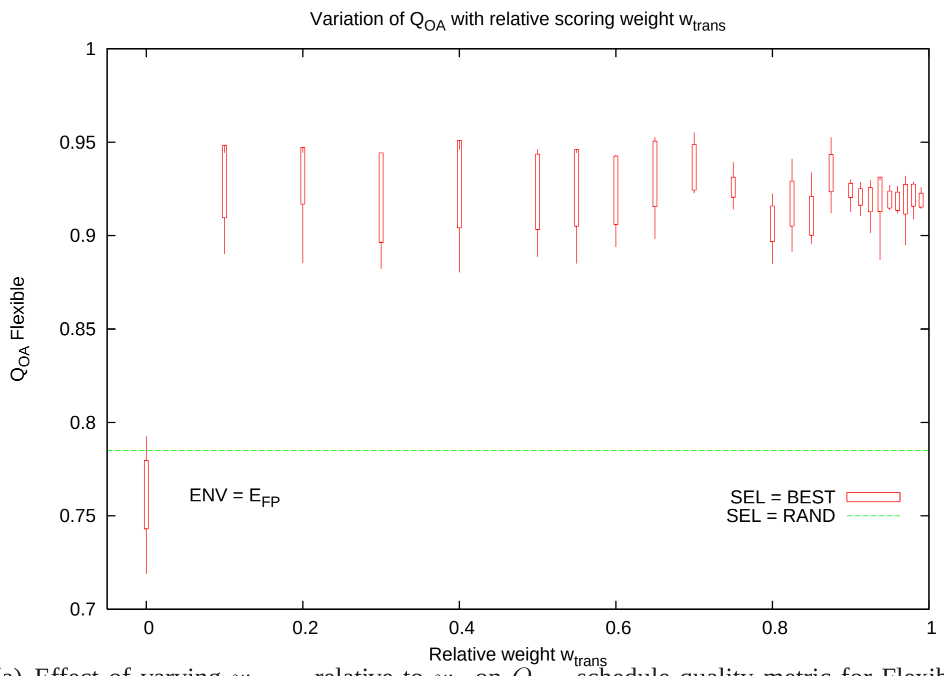


(a) Effect of varying w_{trans} relative to w_p on Q_{OA} schedule quality metric

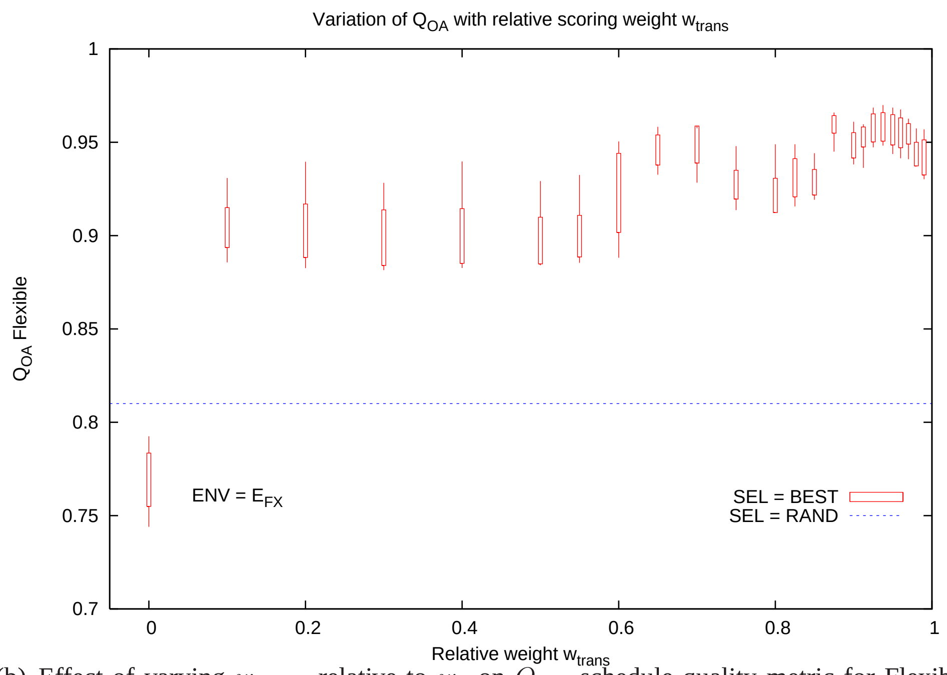


(b) Effect of varying w_{trans} relative to w_p on Q_{OA} schedule quality metric

Figure 8.3: Variation of Q_{OA} with w_{trans} for environment models E_{FP} and E_{FX}.

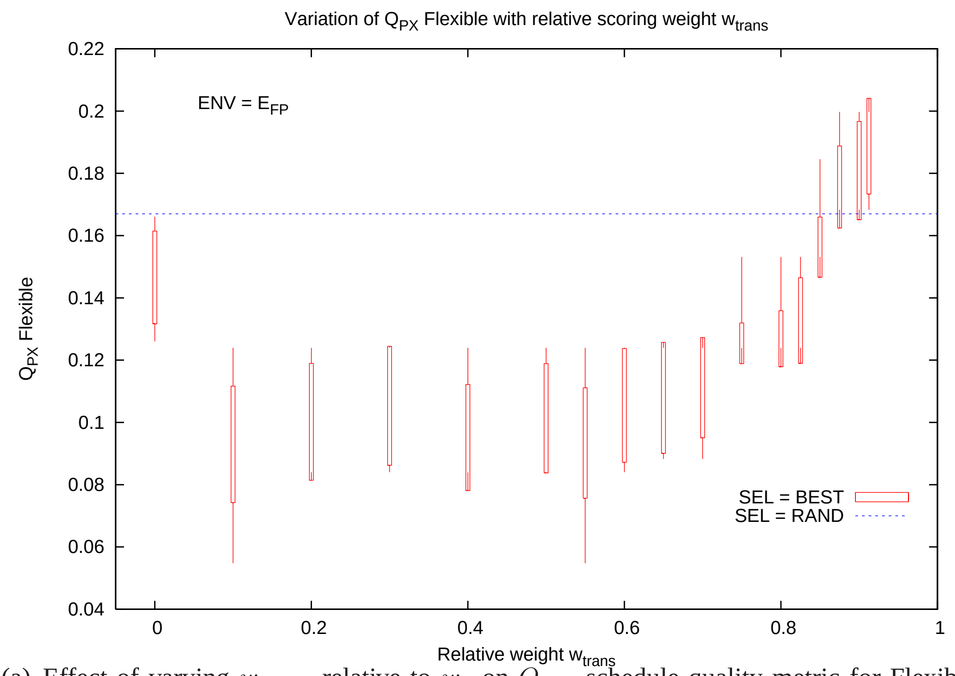


(a) Effect of varying w_{trans} relative to w_p on Q_{OA} schedule quality metric for Flexible groups

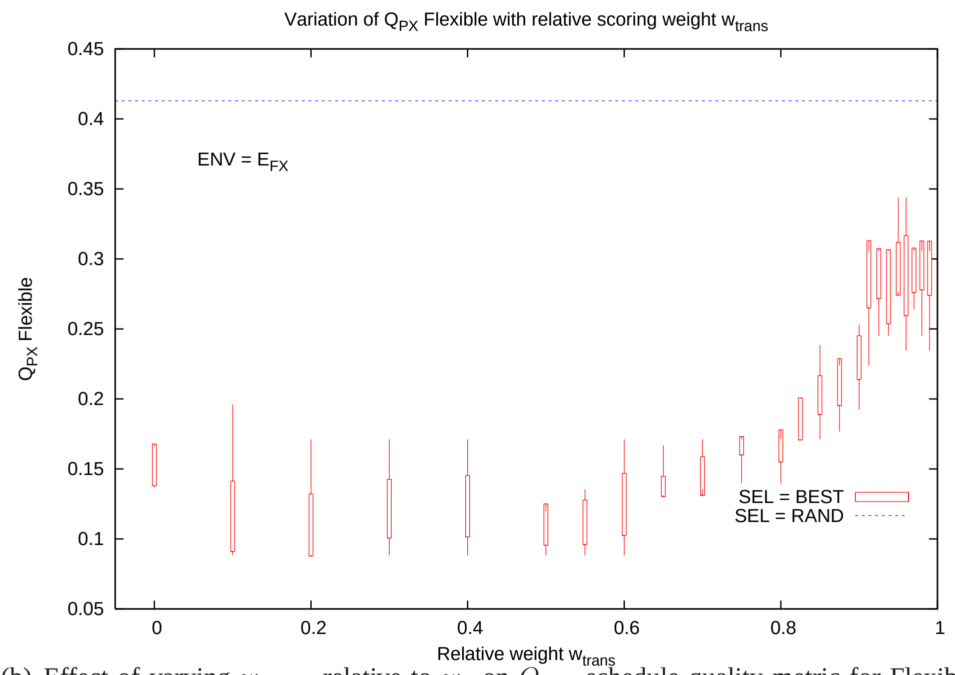


(b) Effect of varying w_{trans} relative to w_p on Q_{OA} schedule quality metric for Flexible groups

Figure 8.4: Variation of Q_{OA} for flexible groups with w_{trans} for environment models E_{FP} and E_{FX}.

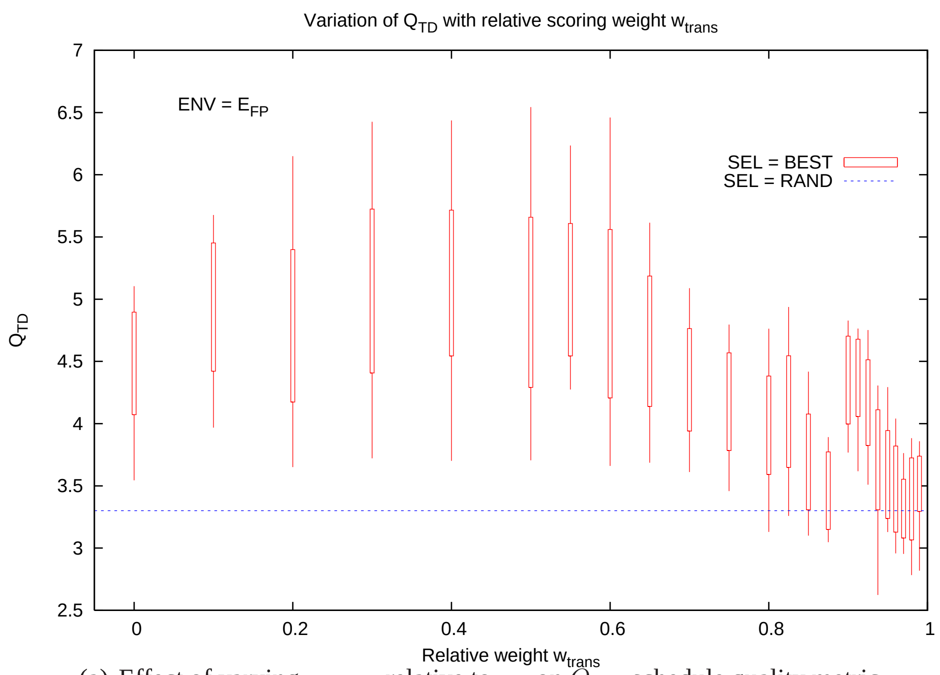


(a) Effect of varying w_{trans} relative to w_p on Q_{PX} schedule quality metric for Flexible groups

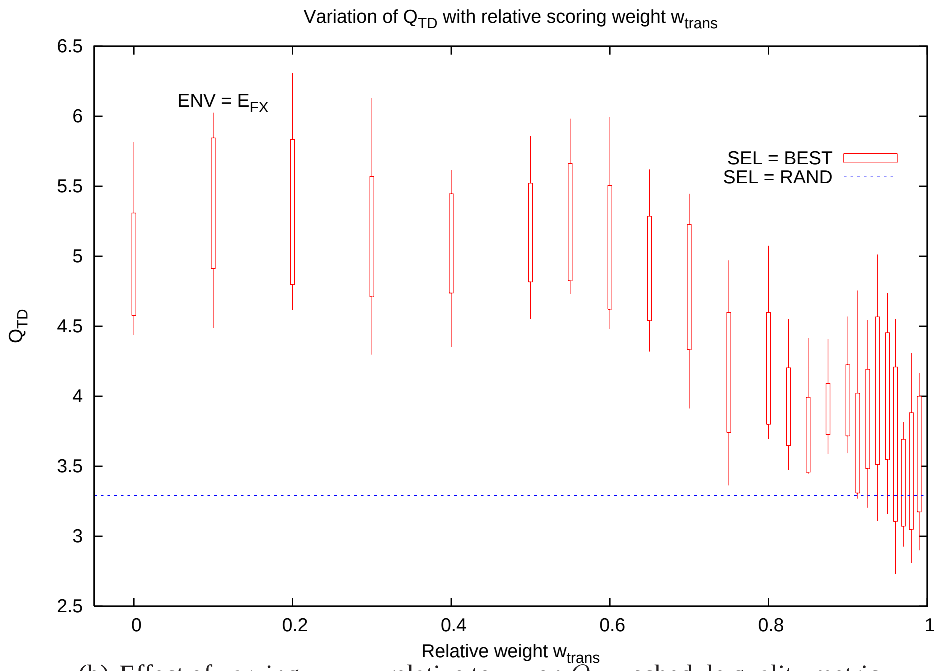


(b) Effect of varying w_{trans} relative to w_p on Q_{PX} schedule quality metric for Flexible groups

Figure 8.5: Variation of Q_{PX} for flexible groups with w_{trans} for environment models E_{FP} and E_{FX}.

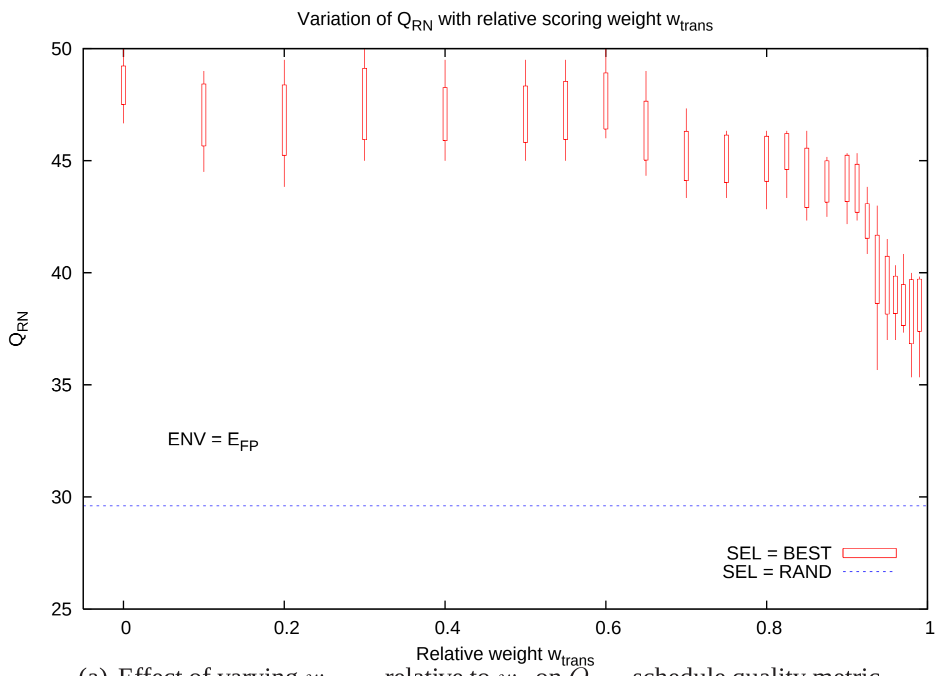


(a) Effect of varying w_{trans} relative to w_p on Q_{TD} schedule quality metric

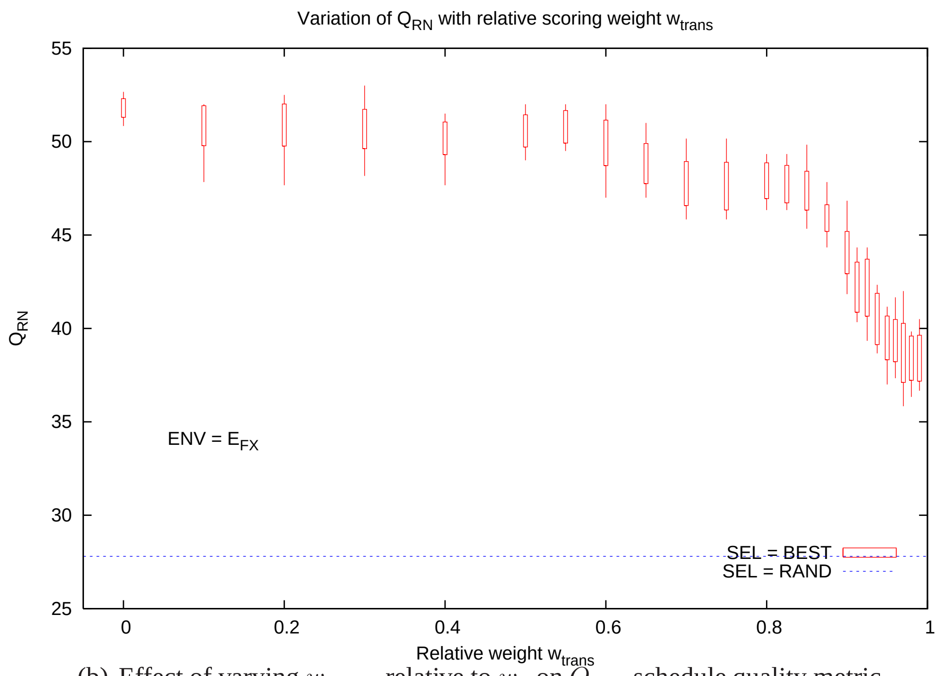


(b) Effect of varying w_{trans} relative to w_p on Q_{TD} schedule quality metric

Figure 8.6: Variation of Q_{TD} with w_{trans} for environment models E_{FP} and E_{FX}.

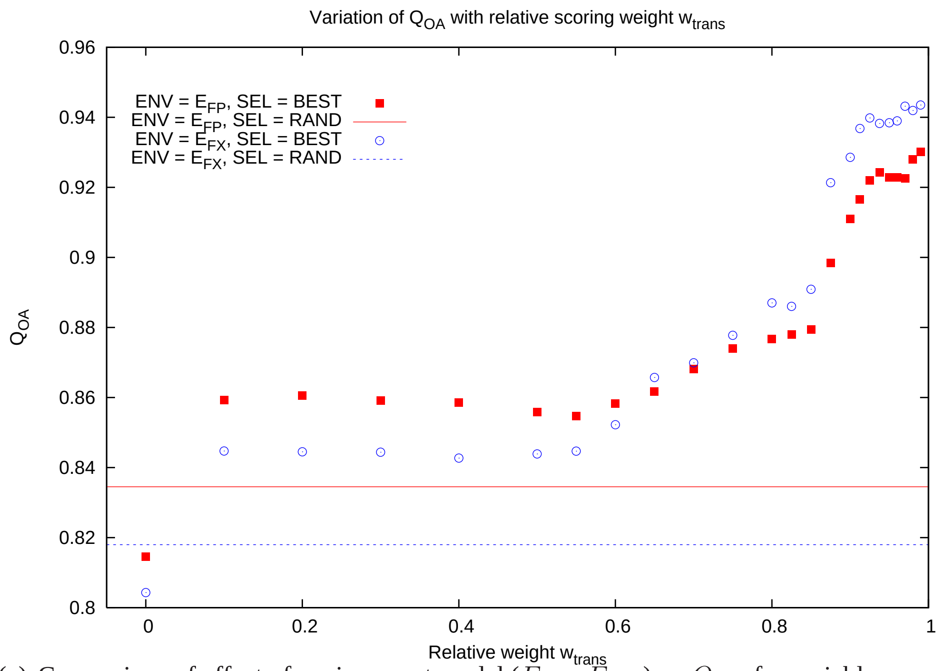


(a) Effect of varying w_{trans} relative to w_p on Q_{RN} schedule quality metric

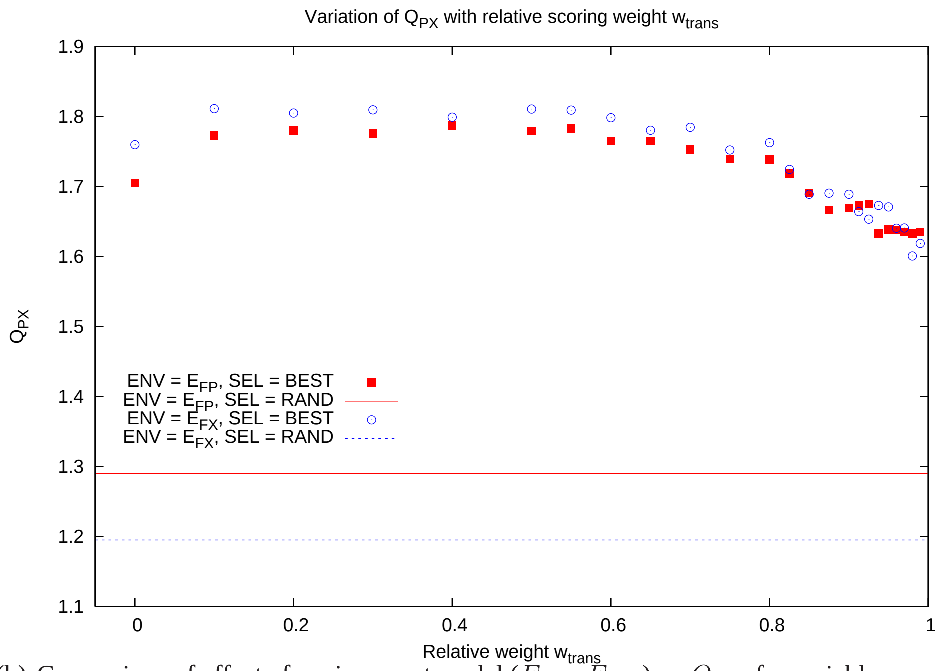


(b) Effect of varying w_{trans} relative to w_p on Q_{RN} schedule quality metric

Figure 8.7: Variation of Q_{RN} with w_{trans} for environment models E_{FP} and E_{FX}.

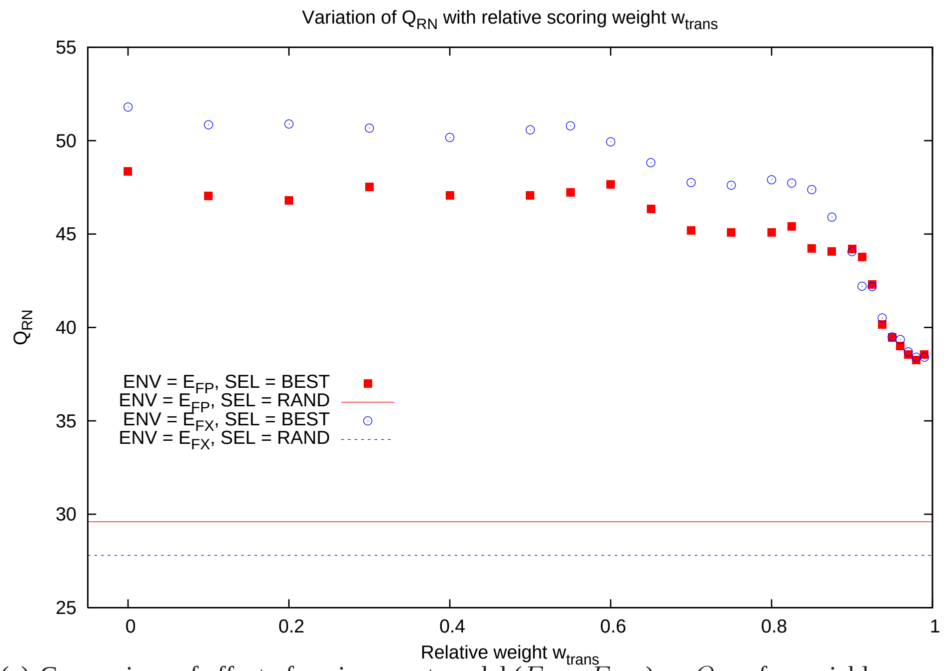


(a) Comparison of effect of environment model (E_{FP}, E_{FX}) on Q_{OA} for variable w_{trans}.

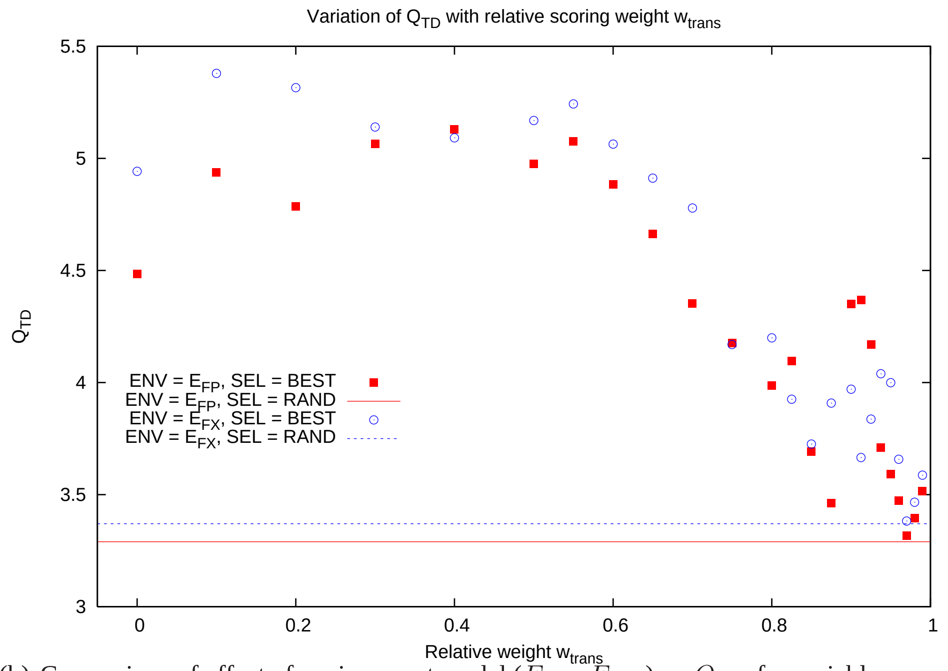


(b) Comparison of effect of environment model (E_{FP}, E_{FX}) on Q_{PX} for variable w_{trans}.

Figure 8.8: Variation of Q_{OA} and Q_{PX} with w_{trans} for variable environment models. Night 1 (27-28 September 2007).

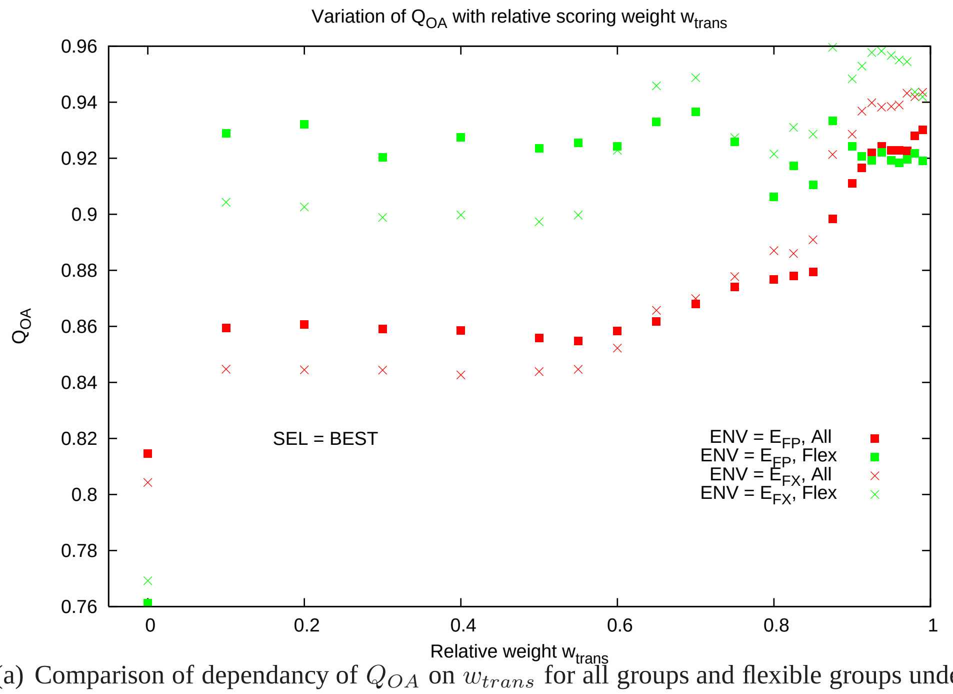


(a) Comparison of effect of environment model (E_{FP}, E_{FX}) on Q_{RN} for variable w_{trans}.

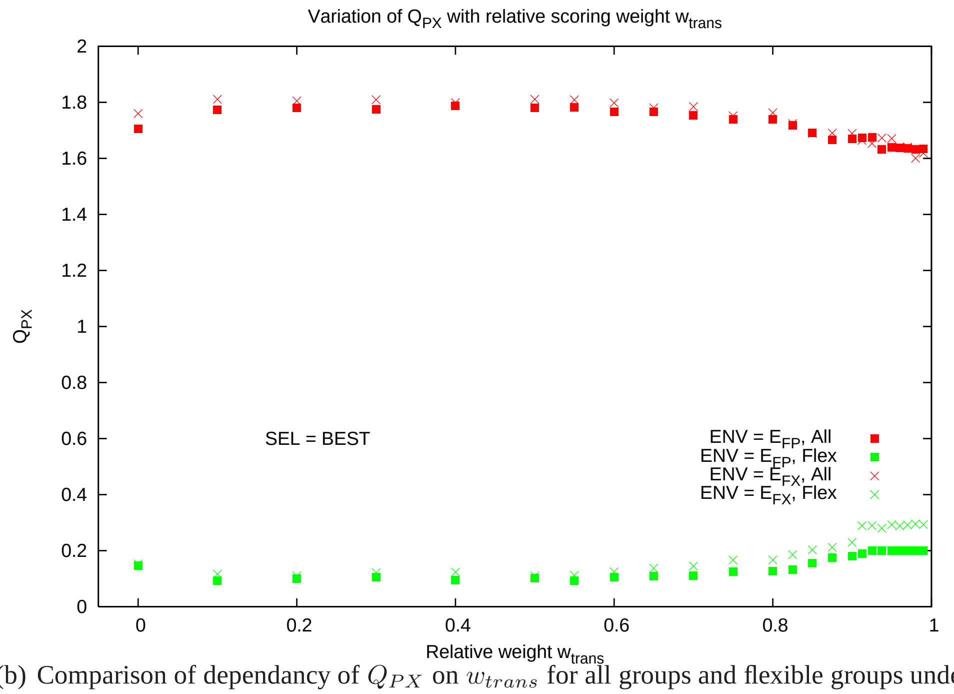


(b) Comparison of effect of environment model (E_{FP}, E_{FX}) on Q_{TD} for variable w_{trans}.

Figure 8.9: Variation of Q_{RN} and Q_{TD} with w_{trans} for variable environment models. Night 1 (27-28 September 2007).



(a) Comparison of dependancy of Q_{OA} on w_{trans} for all groups and flexible groups under environment model E_{FP}.



(b) Comparison of dependancy of Q_{PX} on w_{trans} for all groups and flexible groups under environment model E_{FP}.

Figure 8.10: Variation of Q_{OA} and Q_{PX} with w_{trans} for all groups and flexible groups. Night 1 (27-28 September 2007).

9 Investigation into effects of ODB complexity

9.1 Introduction

Phase 2 models, whether real ODB snapshots or generated are likely to vary somewhat in the range of characteristics measured by the various complexity metrics (PCMs).

The main aims of this investigation are to determine the similarities and differences between real and generated models, the range of variation between generated models created using the same parameters sets and to determine if different schedulers perform better than others when faced with Phase 2 models with different *weight* or *loading* characteristics. It is hoped to be able to answer questions such as whether one scheduler is better at handling *light* loading and perhaps a different scheduler is better at handling *heavy* loading.

9.2 Choice of complexity metric to characterise a Phase 2 model

We have seen from Sect. 3 that there are several complexity metrics (PCMs) available. None of these are able to fully characterize the loading as this varies through the night and from day to day and is dependant on what has (or might have) occurred previously. Of the several contenders, the average contention C_C was chosen. This is a metric which is easy to calculate and readily visualized.

9.3 Comparison of real and generated Phase 2 models

A Phase 2 model generator (P2GEN) was designed to create Phase 2 models with varying characteristics. This generator, described in Appendix A has a large number of configurable parameters. The main parameters in terms of the current study however are the number of proposals, groups per proposal, observations per group and mean

Characteristics of Phase2 models.				
DBID	ODB Date	N_p	N_g	Description
P_s	22/11/07 midpoint	10	10	P2GEN small model
P_l	22/11/07 midpoint	15	20	P2GEN light model
P_m	22/11/07 midpoint	30	30	P2GEN medium model
P_h	22/11/07 midpoint	50	50	P2GEN heavy model
O_1	25/10/07 snapshot	-	-	ODB snapshot Day 0
O_2	15/09/07 snapshot	-	-	ODB snapshot Day -20

Table 9.1: Details of Phase 2 models used in complexity experiments. Generated models P_s through P_h have increasingly higher loading. Models O_1 and O_2 are snapshots taken on first day of the simulations and 20 days prior to the start. Number of proposals (N_p) and number of groups per proposal (N_g) are 2 of the numerous model parameters.

exposure time. The latter determine the length of each group, the others how many groups are in the *pool*. Table. 9.1 summarizes the model and snapshot parameters.

A set of simulations were performed over a period of 60 days for each of the 4 generated models P_l - P_h to guage the level of contention these would yield and to compare with sample ODB snapshots for the same period. The results of a single simulation run for each model are shown in figures 1(a) and 1(b). Similar figures for the ODB snapshots are shown in 2(a) and 2(b).

In addition to variation in characteristics, the real Phase 2 models suffer from a population evolution problem. From Fig. 2(a) which shows the variation of average nightly contention C_C for ODB snapshots taken on the first night of the 60 day period (O_1) and on a night 20 days before the start of the same period (O_2), we see the overall contention seems to decline over the period. We also note in particular that the contention figures for O_2 are generally lower than for snapshot O_1.

These effects are in fact due to the pattern in which observations are entered into the system. A sizeable number of observations are *short-term* and only entered into the ODB a day or so before they are required. In some cases, particularly where automated agents are involved, the observations may be entered minutes before they become activated. This information cannot be captured by a snapshot of the ODB at a given time and so the observed quality and characterisation measures drift away from what would be observed if the ODB model were being regularly replenished.

With a generated model however, groups can all be entered at the start but with their activations spread out over a specified period (say several months) so new groups are appearing in the schedule every day. The day to day loading as shown in Fig. 1(a) appears to remain more constant like that of a real ODB if we were to take daily snapshots.

9.4 Characterisation of generated models

From a given generator model (e.g. P_l) we can generate any number of actual instances. Ideally these would all have the same measurable characteristics (e.g. $\bar{C_c}$) but this is not guaranteed. Further, it is not possible with P2GEN to easily generate a specific Phase 2 model instance with a specific set of PCM characteristics.

Consequently in order to explore the range of these characteristics it is neccessary to generate a number of different models from the same generator parameters and then measure the characteristic of these models either by simulation to obtain dynamic measures or by a straightforward measurement of the static characteristics.

A set of 4 model generators were chosen and are described in more detail in Table. 9.1. These generators are selected to represent a range of characteristics from *small*, designed to create low-load Phase 2 models up to *heavy*, designed to generate high load models. Tests were performed on 25-30 models generated by each generator to allow these to be characterised in terms of the chosen PCM, average contention (C_C). A series of simulations were then performed on each model to generate a set of SQMs, specifically Q_{SU} - a score based metric and Q_{XT} - the fraction of night observed.

The measurable characteristic chosen was $\bar{C_{dc}}$ - the average dynamic contention over the measurement period. A simple scheduler was chosen using best-score selection and a single f_{OH} metric - i.e. the target which was highest relative to maximum attainable elevation was chosen at each sweep. In these simulations we are not particularly interested in the scheduler here, only in the variability of the generated Phase 2 characteristics. Simulations were run for the middle 30 days of the generated models and the values of $\bar{Q_{SU}}$ and $\bar{Q_{XT}}$ are plotted against $\bar{C}_c$ for each model. The results are shown in Fig. 9.3 and Fig. 9.4 with accompanying statistics in Tables B.1 and B.2.

From these results it is clear there is significant variation in the measurable characteristics for any model though there is a progression between models as might be expected i.e. most of the P_l contention values are lower than most of the P_h values. C_C varies from 2.69± 1.02 for P_S upto 18.8± 3.5 for P_H. There is significant overlap between *adjacent* models. The heavy model generally uses up all or very nearly all of the available night(95% ± 3.8%) whilst the light model fills only 85% ± 5% of the night. This is not too surprising, there are more groups to chose from in the heavy model so there are likely to be few if any slack periods. There is most variation in Q_{SU} for the small and light models (46 $\pm$ 7.4) for P_S while the heavy model achieves a typical score of 66 $\pm$ 3.6. With low contention there will be periods when no groups are actually schedulable hence the frequent depression of this metric and in the overall use of time. The data from Fig. 9.3 were fitted with a least squares fit of the form $Q_{SU} = a + bC_C$ which after 5 iterations gave a fit with $a = 44.2 \pm 2$% and $b = 1.124 \pm 6$% and $\chi^2 = 23.32$.

9.5 Comparison of schedulers against variable loading characteristics

In order to establish the effectiveness of different schedulers against ODB loading a number of simulations were performed. In order to test a suggestion from Cicirello and Smith (2002) that introduction of a degree of randomization into the selection process could increase overall reward, the BDS scheduler was setup using 3 different selection models:-

- *Best* selection ζ_{Best} is the selection model normally employed in which the highest scoring group is selected.
- *Fixed rank* selection ζ_{FR} allows the second, third etc ranked group to be selected with a fixed probability. In the current experiment these probabilities were set to: rank 1 (80%), rank 2 (15%), rank 3 (5%).
- *Rank scaled* selection ζ_{RS} assigns a probability to each of the top ranked groups in proportion to the groups actual score. This is based on the observation that the top ranked groups often score very similar values implying that that they are more or less equally deserving of execution.

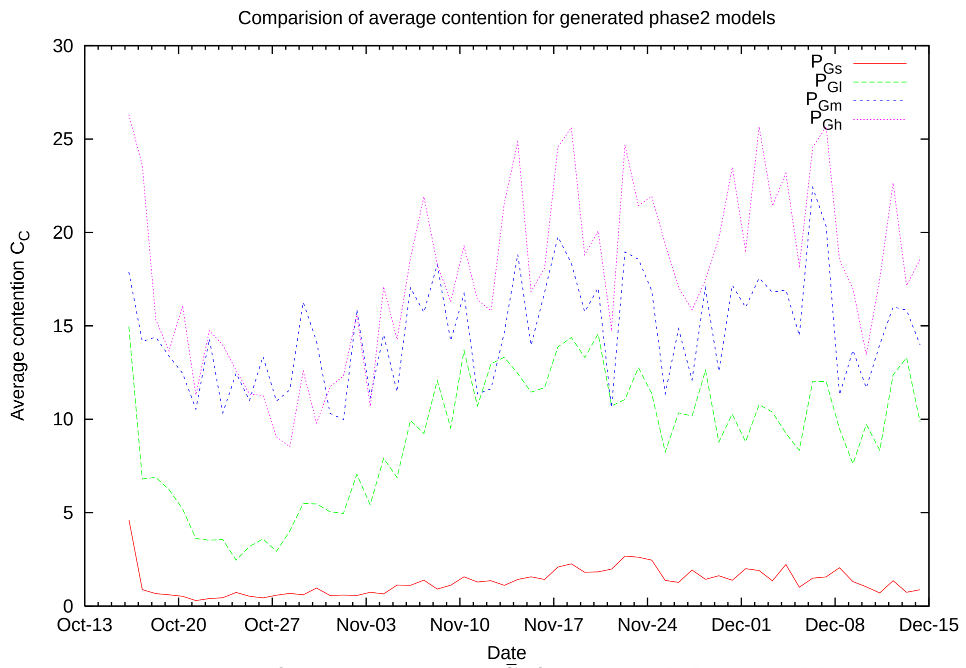
The look-ahead scheduler QLAS was tested using horizons of 1,2 and 4 hours.

Results for the BDS simulations are shown in Figs. 9.5, 9.6 and 9.7. There is clearly a significant degree of variation between individual runs at the different Phase 2 loads though this is expected from the earlier results in Sect. 9.4. The only noteable feature appears to be that *fixed rank biased* selection model tends to have a greater degree of variation than *best* and *rank score biased* selection.

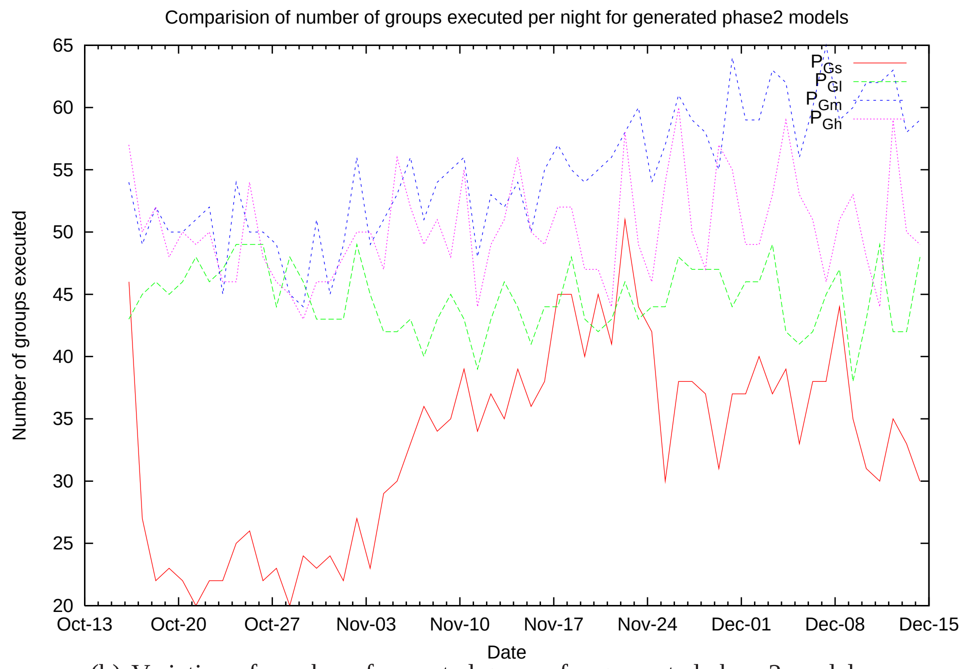
Statistics for these results are shown in Table B.19. For QLAS, Figs. 9.8, 9.9 and 9.10 show that there is again significant variation from run to run at the different loads. However the general trend is for an improvement in scoring as the horizon is increased. Fig. 9.11 shows a comparison in which *best-fit* polynomial curves were constructed using least squares methods for the results from the BDS and QLAS schedulers. These show a clear but small improvement in average quality with increasing horizon (from 6.3% ± 10 for QLAS with H=1 to 19% ± 8 for QLAS with H=4) and that BDS performs as well or better in low load situations. We also see a modest average improvement in quality at higher loads over the normal BDS ζ_{best} selection by BDS with ζ_{RS} selection.

9.6 Summary and conclusions

I have established that it is possible to simulate Phase 2 characteristics by using suitable generator models but that it is not feasible to create a Phase 2 model with specific characteristics. If such a model were required it would be neccessary to create multiple instances and test these until a suitable candidate were found. There is significant variation in the characteristics of generated Phase 2 models but these do roughly scale with the generator model's *weight*. When using real Phase 2 models (ODB snapshots) it is important to realise that a significant fraction of the observations are entered with short lead times and so are not visible in the snapshot a few days in advance. The charcateristics of the snapshot will depart from a snapshot taken a few days later by a significant amount. With generated models this effect can be mitigated against by ensuring that observations are entered with longer lead times. The experiments showed that at low levels of complexity there is considerable variation in the fullness (Q_{XT}) of schedules generated by all scheduling paradigms. As the level of complexity increases the degree of variation decreases. There is some evidence that longer look-ahead horizons are suitable for higher density ODBs and can produce better overall schedules. There is also slight evidence that biased selection can improve BDS scores at higher loads.

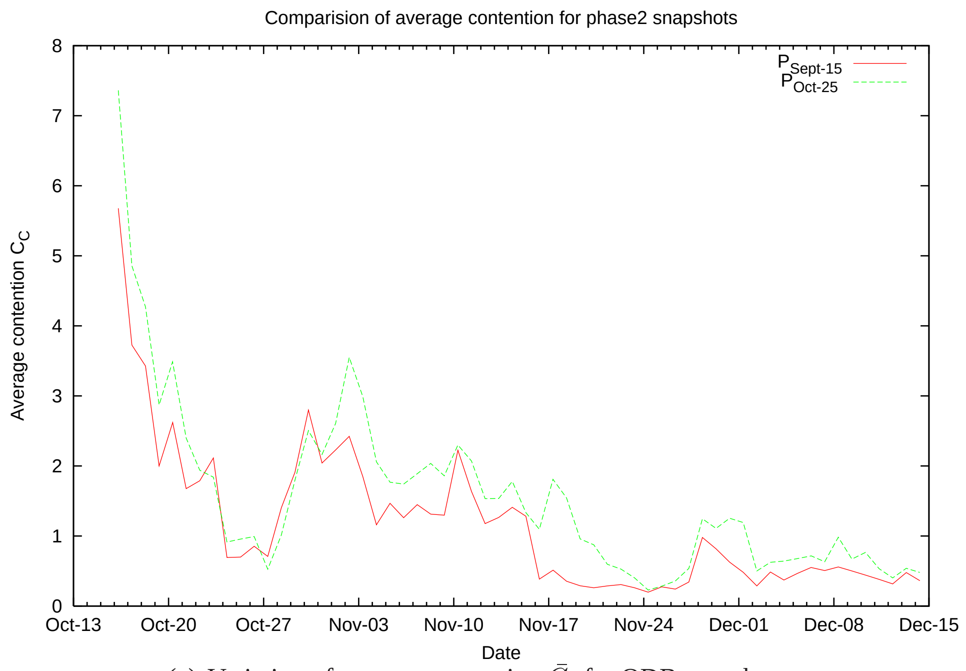


(a) Variation of average contention $\bar{C}_c$ for generated phase2 models.

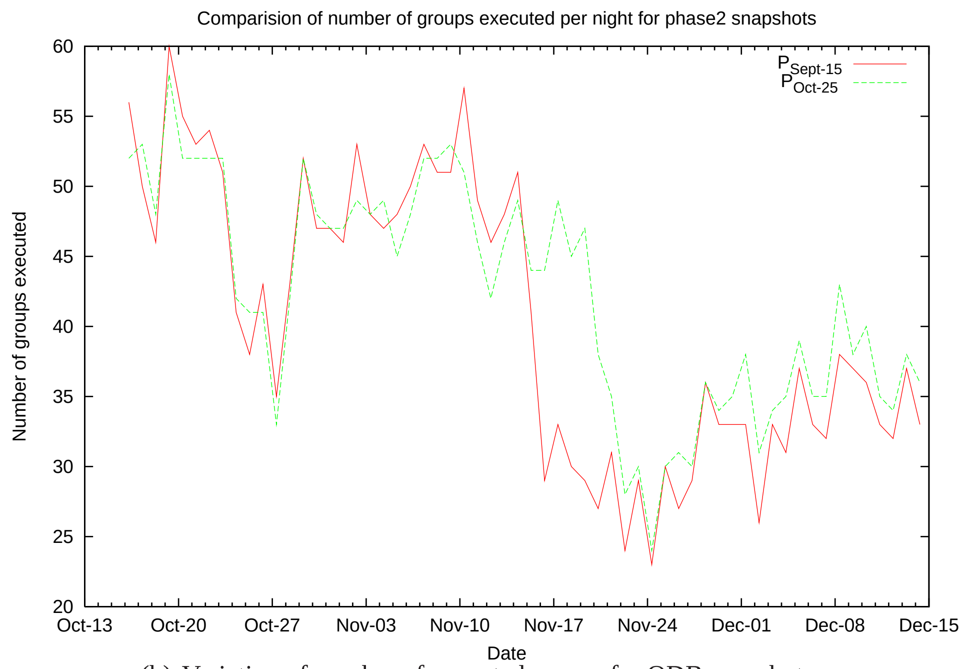


(b) Variation of number of executed groups for generated phase2 models.

Figure 9.1: Comparison of average contention measure and number of groups executed per night for generated Phase 2 models.



(a) Variation of average contention $\bar{C}_c$ for ODB snapshots.



(b) Variation of number of executed groups for ODB snapshots.

Figure 9.2: Comparison of average contention measure and number of groups executed per night for ODB snapshots.

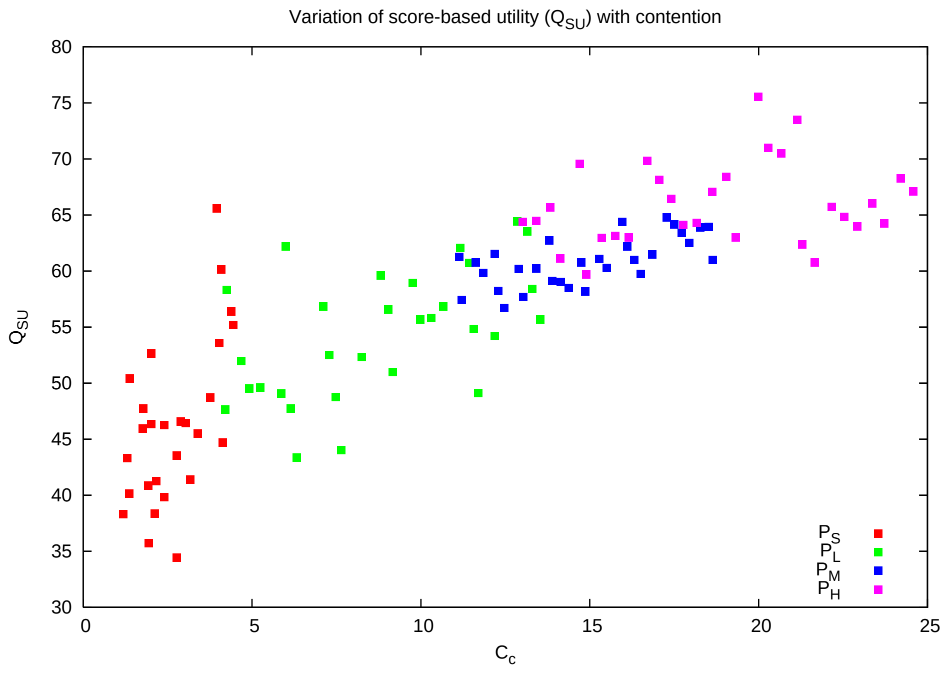


Figure 9.3: Variation of Q_{SU} with C_{DC}. Each point represents a single phase 2 model generated by one of 4 initial sets of generators.

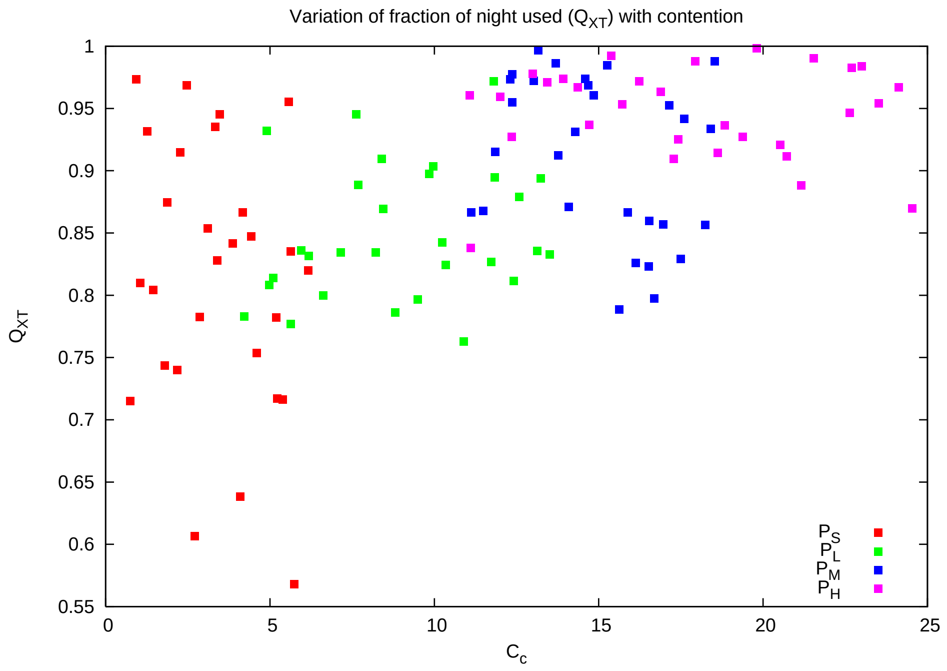


Figure 9.4: Variation of used fraction of night Q_{XT} with C_{DC}. Each point represents a single Phase 2 model generated by one of 4 initial sets of generators.

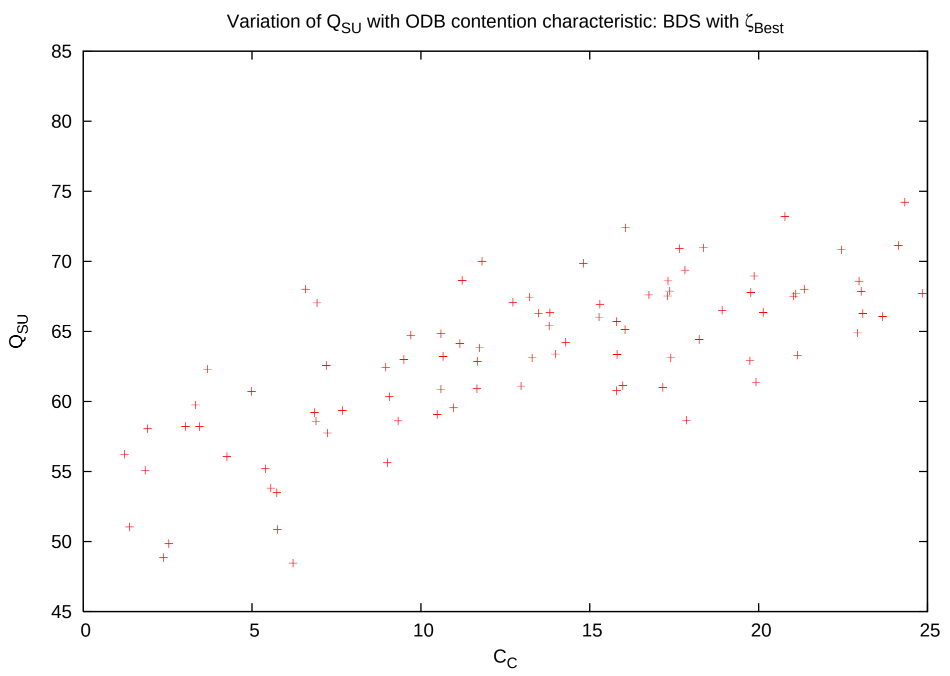


Figure 9.5: Effect of selection model on variation of Q_{SU} with C_c for BDS using ζ_{Best}. There is significant variation between runs.

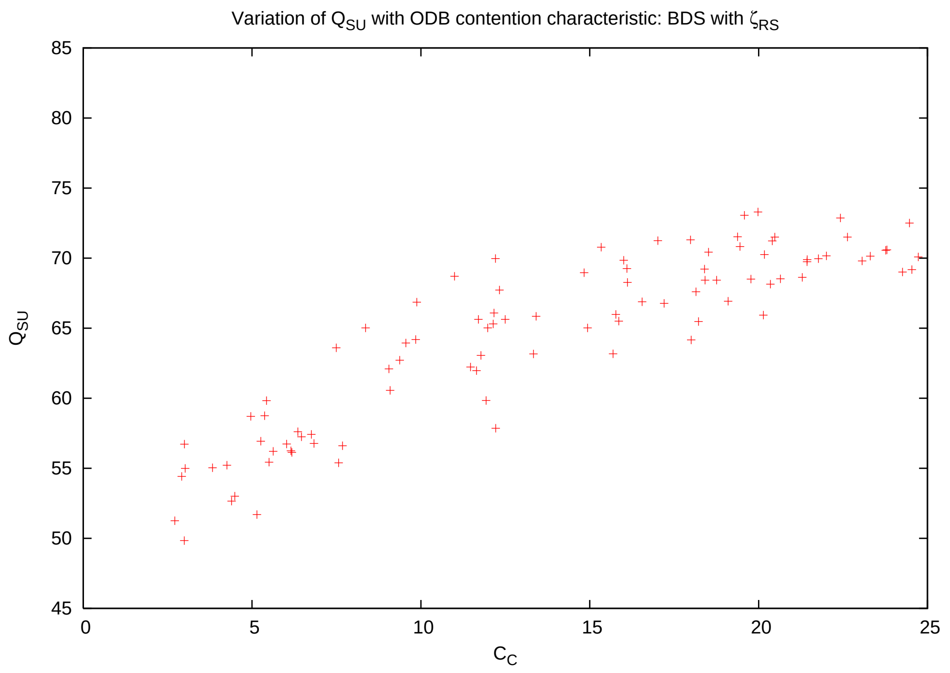


Figure 9.6: Effect of selection model on variation of Q_{SU} with C_c for BDS using ζ_{RS}.

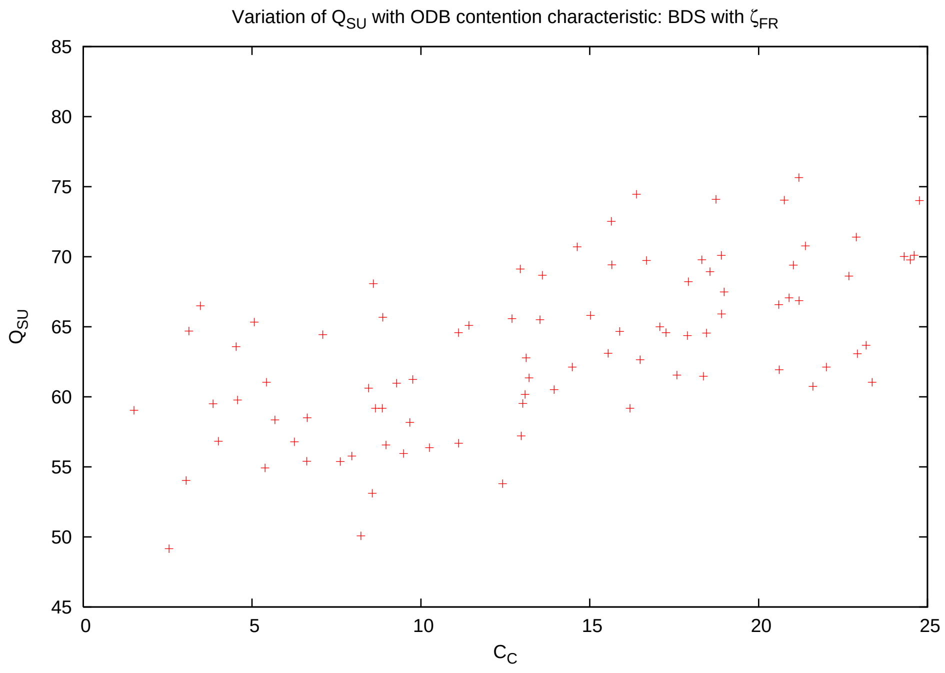


Figure 9.7: Effect of selection model on variation of Q_{SU} with C_c for BDS using ζ_{FR}.

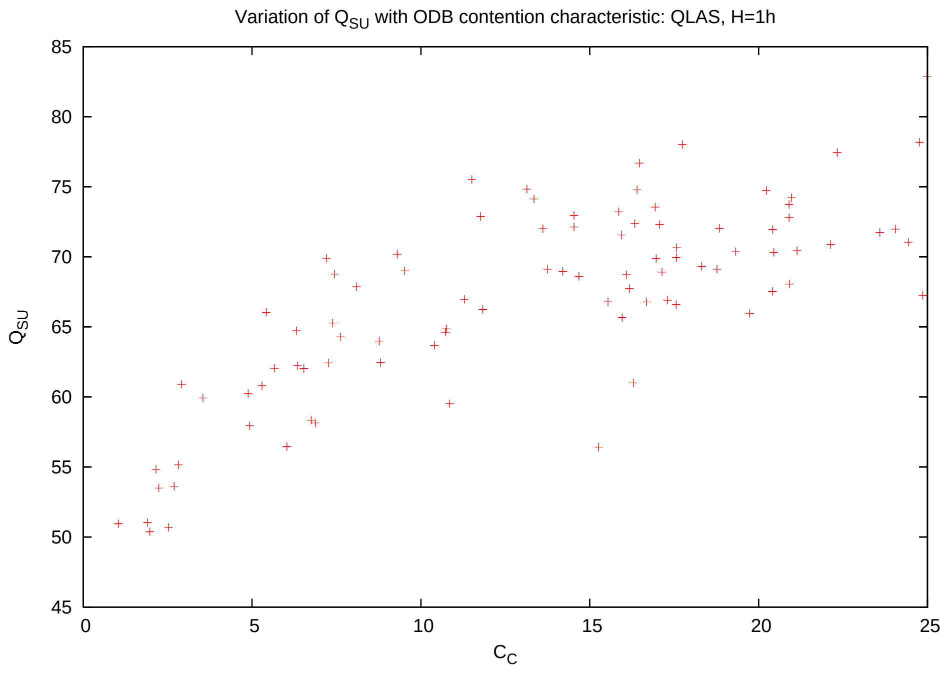


Figure 9.8: Effect of horizon on variation of Q_{SU} with C_c for QLAS with $H = 1$h.

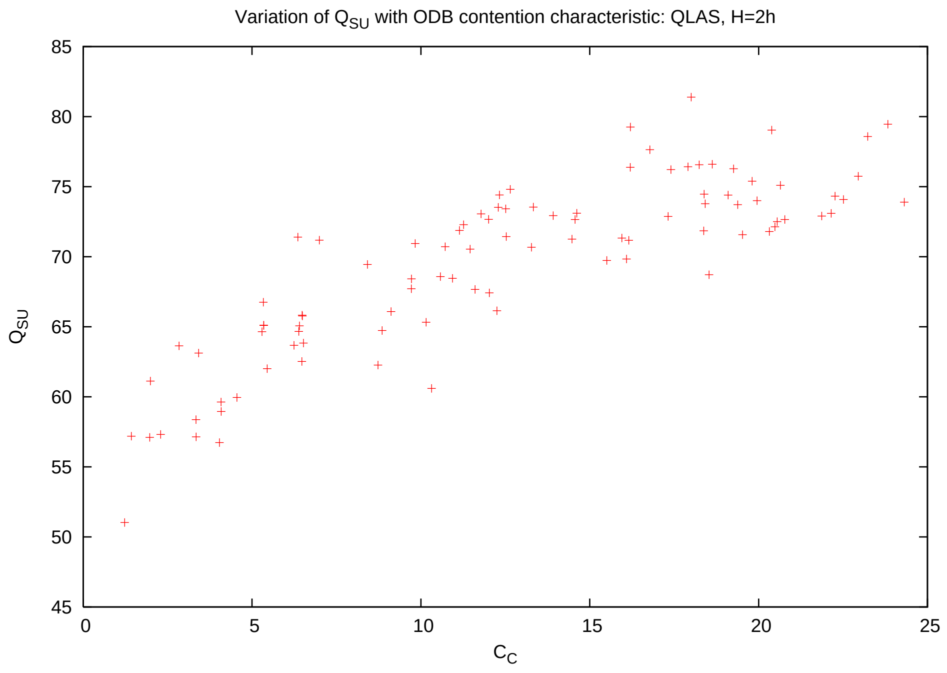


Figure 9.9: Effect of horizon on variation of Q_{SU} with C_c for QLAS with $H = 2$h.

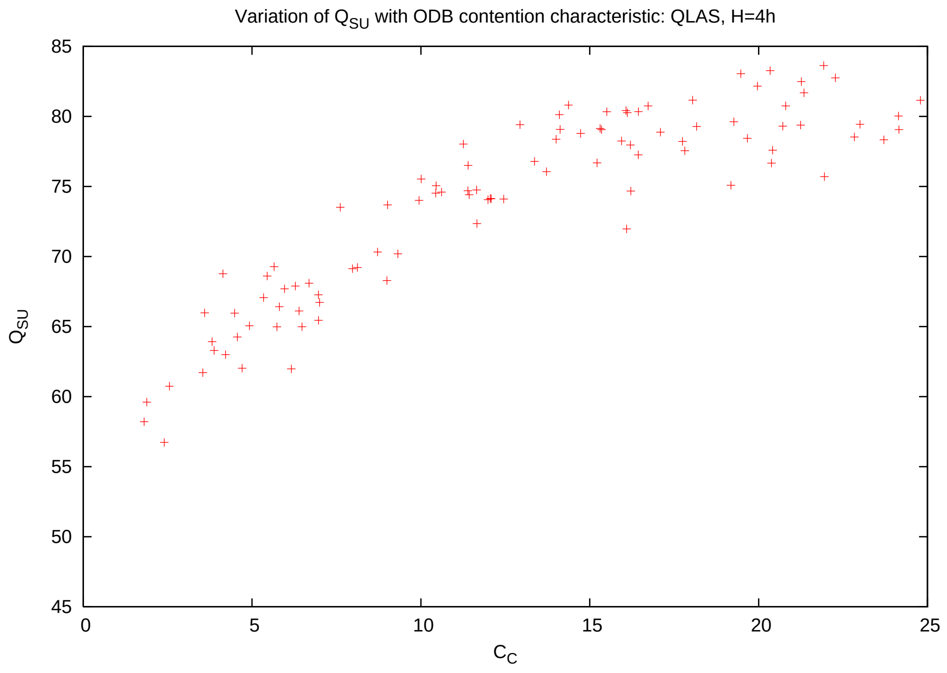


Figure 9.10: Effect of horizon on variation of Q_{SU} with C_c for QLAS with $H = 4$h.

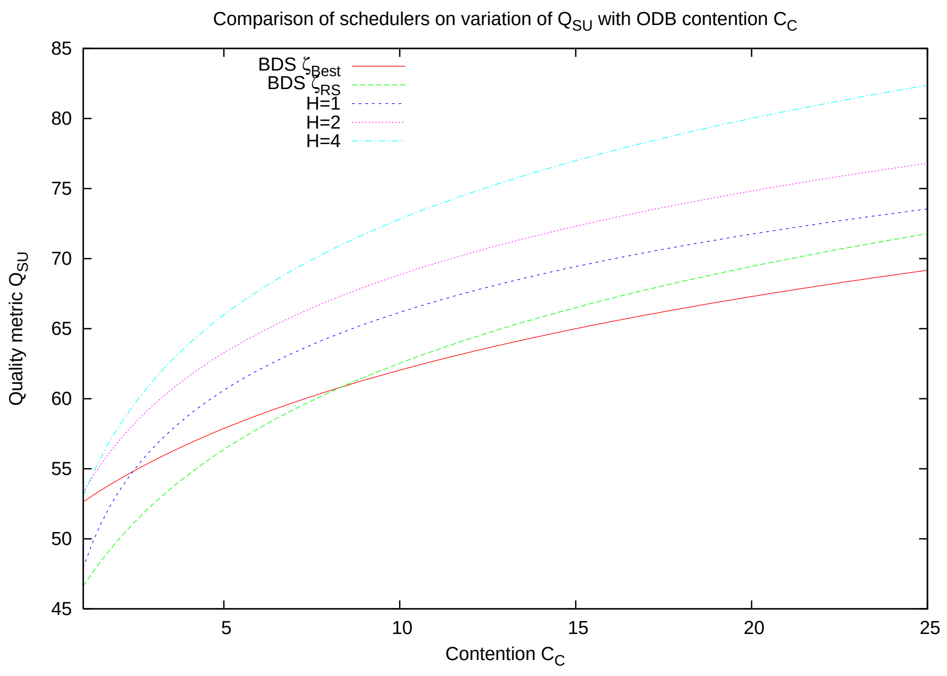


Figure 9.11: Effect of choice of scheduler on variation of Q_{SU} with C_c. The curves are polynomial least squares fits to the data from Figs. 9.8 through 9.10 for BDS and QLAS.

10 Investigation into effects of environmental stability

10.1 Introduction

This series of experiments is aimed at comparing the effectiveness of different scheduling algorithms and configurations under varying conditions of environmental stability over an extended period. A secondary aim is to test the environment model generator to determine the range of characteristics which are generated for a given set of initial parameters.

10.2 Modelling the environment

From section 4.4.1 we see that the environmental conditions at the telescope site (specifically seeing) vary over different timescales from minutes to hours. From the point of view of scheduling, the seeing conditions s are split into several bands defined as: (G) good ($s < 0.8''$), (A) average ($0.8'' < s <= 1.3''$), (P) poor ($1.3'' < s <= 3.0''$) , (U) usable ($3'' < s <= 5''$), (B) bad/unusable ($s > 5''$). So long as the seeing varies only within any band, neither contention or scoring of groups are affected. When the seeing transitions between bands, both are affected.

When running a simulation, one option for the environment model is to specify the sequence of environment changes over the simulation duration. Such an *environment scenario* defines when and how the seeing changes between bands. This is most useful when we want to be able to repeat a particular simulation step where some other parameter of interest is being varied.

From a defined set of environment model generator parameters we can generate any number of different scenarios. It would be useful to see if we get variation in results and indeed in the measured SQMs from different scenarios generated by the same initial model parameters.

Seeing variation in this series of experiments is modelled by a simple transition matrix $P(t)$ where each element $P_{ij}(t)$ represents the cumulative probability of transitioning from state i to state j within a period t. The diagonal elements $P_{ii}(t)$ represent the probability of *remaining* in a particular state for time up to t.

$$\begin{pmatrix} P_{gg}(t) & P_{ga}(t) & P_{gp}(t) & P_{gu}(t) & P_{gb}(t) \\ P_{ag}(t) & P_{aa}(t) & P_{ap}(t) & P_{au}(t) & P_{ab}(t) \\ P_{pg}(t) & P_{pa}(t) & P_{pp}(t) & P_{gu}(t) & P_{pb}(t) \\ P_{ug}(t) & P_{ua}(t) & P_{up}(t) & P_{uu}(t) & P_{ub}(t) \\ P_{bg}(t) & P_{ba}(t) & P_{bp}(t) & P_{bu}(t) & P_{bb}(t) \end{pmatrix} \tag{21}$$

The relative distribution of time spent in each band is of less interest from the point of view of these studies. Only the stability or frequency of change is important. The diagonal elements were therefore set to the same value based on a stability parameter τ_E such that the probability of remaining in a given band/state for up to time t is given by $P_{ii}(t) = 1 - \exp(-t/\tau_E)$. The *bad* and *usable* bands are also of less interest. Bad indicates no observing possible, and in practice relatively few groups are ever setup with such lax seeing constraints. Consequently the array is reduced to 3x3. The non-diagonal elements are each set to the same value, namely $P_{ij} = 0.5 * (1 - P_{ii})$. The parameter τ_E which characterizes the scenario is an indicator of stability. When τ_E is small, the environment is unstable with changes between bands occurring frequently. With large values of τ_E, the environment is stable and changes occur only rarely.

A specific seeing scenario is generated by taking random intervals calculated from $-\tau_E \ln(1 - R)$ where R is a random number in $[0, 1]$. At each period the seeing is allowed to change randomly to another value (band). We are not particularly interested as to which band it change or how the time spent in the different bands is distributed, only how frequently it is changing.

Using an environment model generator with a stability time set to 30 minutes (1800 sec) a number of runs were made to generate scenarios over a period of 60 days. The results of 4 of these are plotted in Fig. 10.1 which shows the distribution by time of the lengths of stable periods. The distributions peak around 30 minutes (average for the 4 runs shown is 31.4 minutes) and there is significant small scale variation from run to run. With 30 minutes average period and 60 day runs I would expect around 3000 total samples divided into 50 bins giving around 60 samples per bin. This would give random errors per bin of around 12% which agrees well with the variations shown.

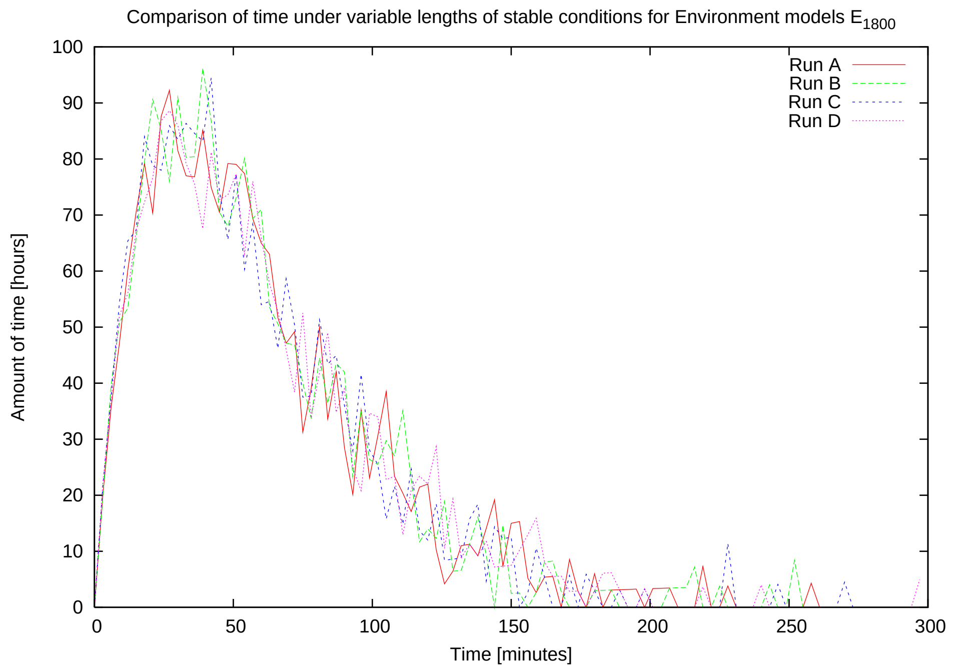


Figure 10.1: Comparison of relative amounts of stable time between state transitions for several runs of environment scenario E_{1800}. The average length of stable periods is 30 minutes. There is significant variation between runs at a finer scale amounting to around $\pm 12\%$ per bin in agreement with expected poisson errors.

10.3 Shakedown experiments

The two schedulers previously discussed (BDS and QLAS) were compared under varying conditions of environmental stability. BDS was tested using the 3 selection models described in Sect. 9.5. An initial test was made using an environment scenario generator set to produce a number of different highly unstable scenarios with $\tau_E = 0.5$ hours. The Phase II model employed produces a *medium* load and in this case was skewed to have a high percentage of priority observations requiring *dark moon* conditions, resulting in pronounced monthly variations in most quality metrics (see dips around 20^{th} May and 18^{th} June. A normal stochastic execution timing model was used resulting in some random variation of group execution durations. The results for the various BDS configurations are shown as ensemble plots in Figs. 10.2 through 10.4.

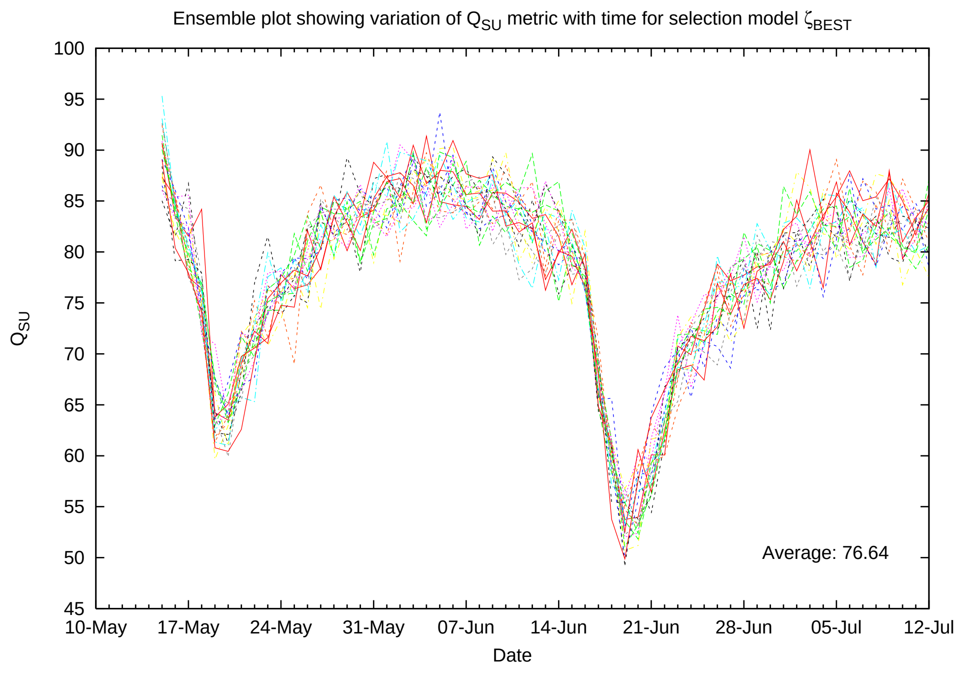


Figure 10.2: Ensemble plot showing variation of Q_{SU} with time for selection model ζ_{Best}. Peak value which occurs around new moon when more targets are available is 85 with minimum at full moon of 50.

The quantity plotted is the score based metric Q_{SU} computed for each night using the scores f_{SU} of each selected group weighted by its duration $\sum_i f_{SUi} X_i$. The plots all have a similar look in that the measured metric is seen to vary between lower and upper bounds over the course of the approximately 2 lunar cycles. Peak values occur during new moon when the higher priority groups are more prominent and lower values during full moon when these are reduced.

The *Best* selection would appear to produce the least variation between runs in that the plots are fairly tightly enveloped. *Fixed-rank* selection produces on average a slightly higher score during new moon but during the first full moon its results are poorer. It has the least tightly enveloped of the plots. This implies that this model has

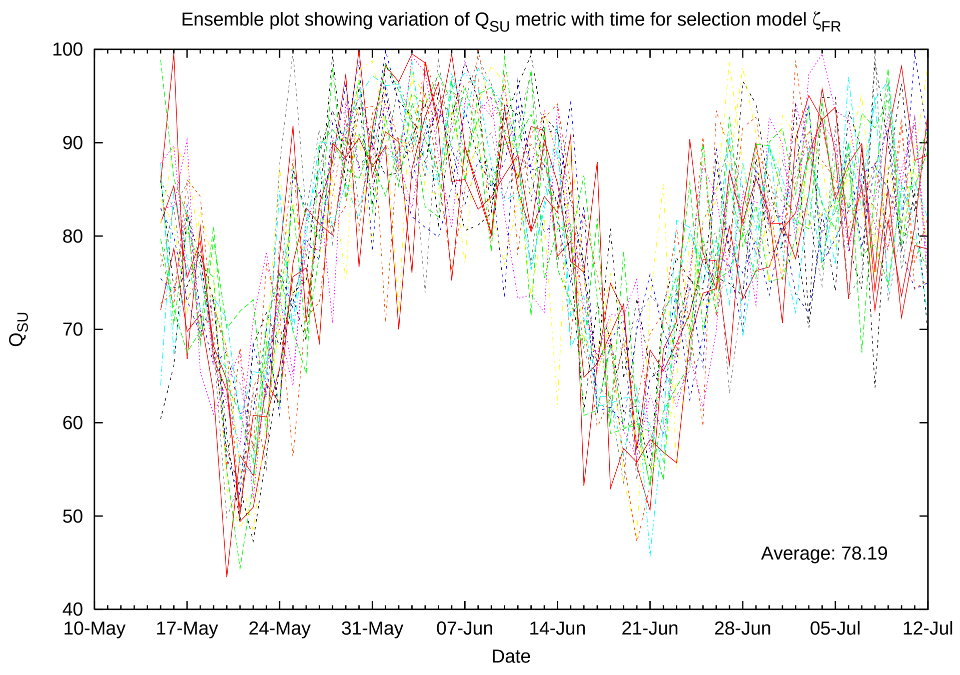


Figure 10.3: Ensemble plot showing variation of Q_{SU} with time for selection model ζ_{FR}. Peak value is 90 with minimum 60. There is significantly more variation between runs than for ζ_{Best}.

the potential for better results but a significant chance of poorer results due to the extra degree of variation introduced. *Rank-scaled* selection appears to produce on average the most increase in reward though again there is significantly more variation from night to night than *best* selection. Tables B.3 and B.4 contain some statistics derived from these ensembles.

Table B.3 shows the fraction of nights on which the biased selection models managed to beat (in terms of score) the baseline scheduler (BDS + ζ_{best}) as well as the fraction of nights on which the baseline scheduler won. It can be seen that overall the biased models performed slighly better (e.g. ζ_{RS} beats ζ_{best} on 62% of nights while ζ_{best} beats ζ_{RS} on only 38% of nights. Table B.4 shows the amount (score differential) by which each model beats its opponents. We see that e.g on those nights when ζ_{RS} beats ζ_{best} it does so by 8.53 whereas when ζ_{best} beats ζ_{RS} it only does so by 5.76. A typical nightly score for any model is between 60 and 90 so these differentials are of order 6% to 14% of the total score.

Results for QLAS are shown in Figs. 10.5 and 10.6. These plots show typical ex-

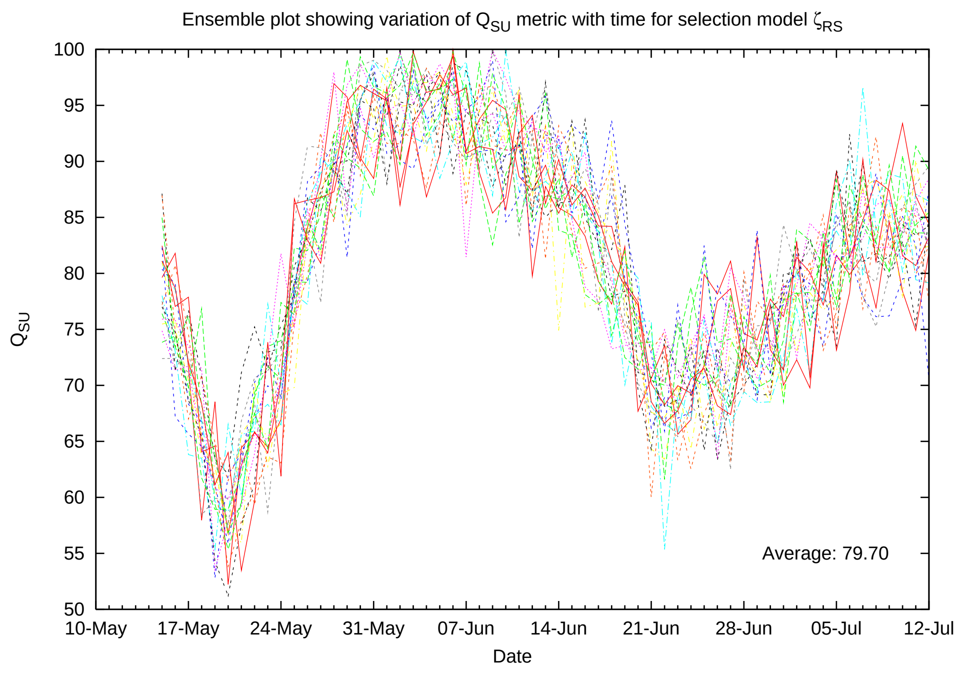


Figure 10.4: Ensemble plot showing variation of Q_{SU} with time for selection model ζ_{RS}. peak values are 95 with minimums of 70. This model seems to produce the highest improvement but with significantly more variability than ζ_{Best}.

ample runs for each QLAS horizon and a typical BDS run for comparison. Fig. 10.5 shows Q_{XT} the total fraction of the night observed. It seems clear that BDS frequently manages to fill the night better than QLAS for this unstable situation. The reward (Q_{SU}) plotted in Fig. 10.6 also shows that in this scenario, BDS is capable of holding up well to QLAS.

10.4 Detailed experiments

The BDS configurations and QLAS were then tested using a range of environment scenarios with τ_E ranging from 0.5 hours up to 4 hours. The same scoring model was used for both BDS and QLAS and the same MD1 phase 2 model as the earlier experiments. The stochastic execution timing model was swapped for a fixed timing model to reduce any extraneous variation other than that introduced by the environment scenarios. For each scheduler configuration a total of 500 simulations were run over the 60 day period.

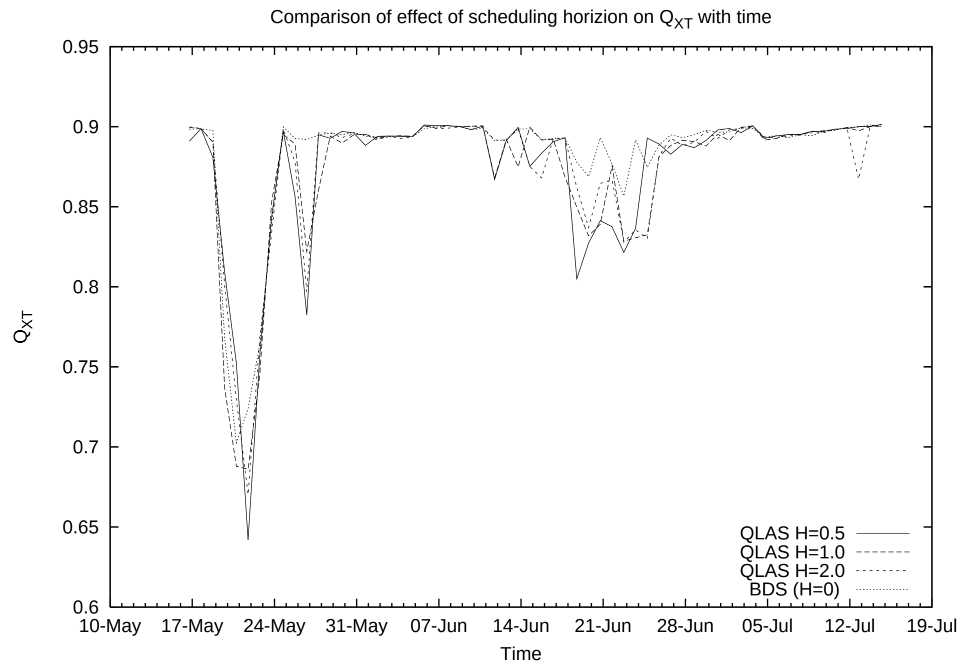


Figure 10.5: Variation of Q_{XT} with time for BDS and QLAS with horizons 0.5, 1 and 2 hours. During the lunar cycle dips around 18 May and 21 June BDS holds up well against QLAS.

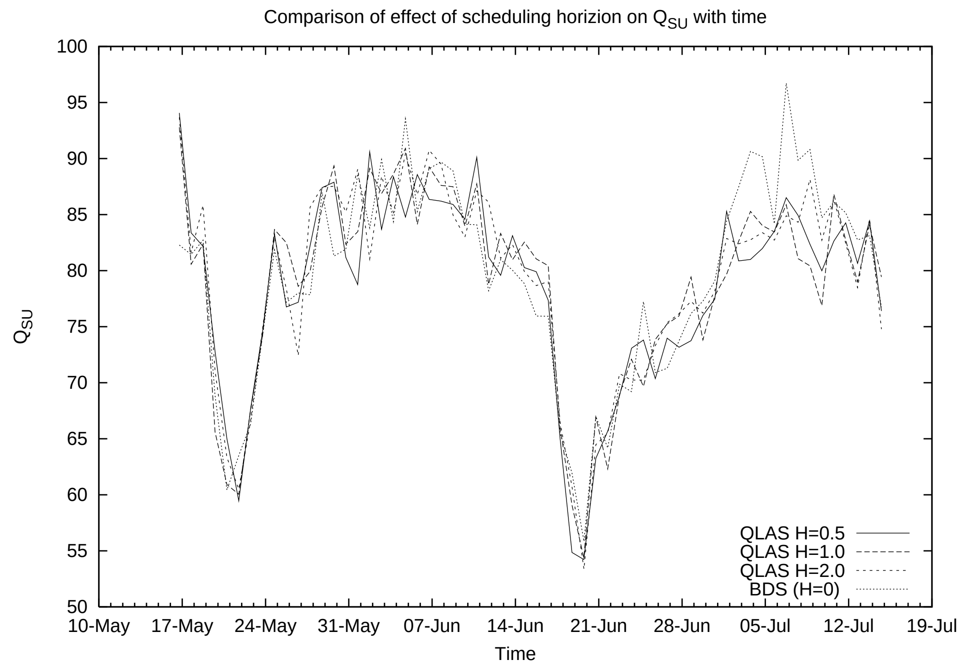


Figure 10.6: Variation of Q_{SU} with time for BDS and QLAS with horizons 0.5, 1, and 2 hours. BDS scheduler performs well against QLAS particularly during second lunar peak around 21 June.

The results of the experiments for BDS are shown in Figs. 10.7 through 10.9 with accompanying statistics for selected values of τ_E in Table. B.20. In each case the measured metric is again Q_{SU} averaged over the full run. There appears to be little to chose between the different BDS selection models. Fig. 10.9 does suggest that ζ_{RS} may offer an improvement over ζ_{Best} at least under unstable conditions, though this seems less clear as stability improves. The results for ζ_{FR} Fig. 10.8 show marginally more variability than the other 2 selection models with possibly a hint of improvement at low τ_E.

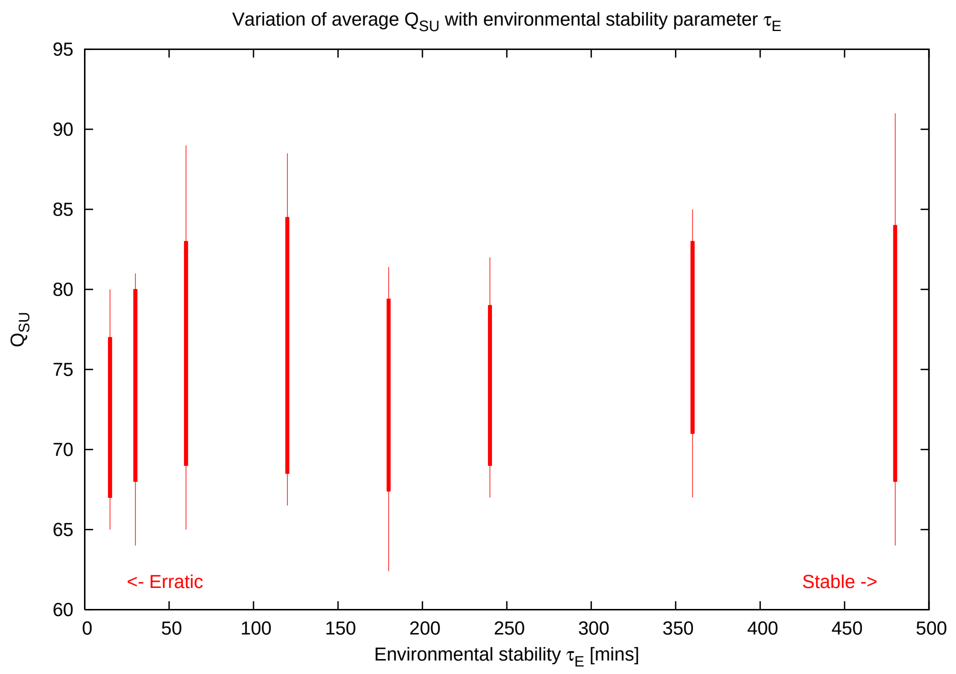


Figure 10.7: Variation of Q_{SU} with τ_E for selection model ζ_{Best}. Significant variation between runs, but little variation with τ_E.

For the QLAS, seperate tests were performed to determine the effects of varying the length of the look-ahead horizon H and the QLAS sequence count control parameter N_s.

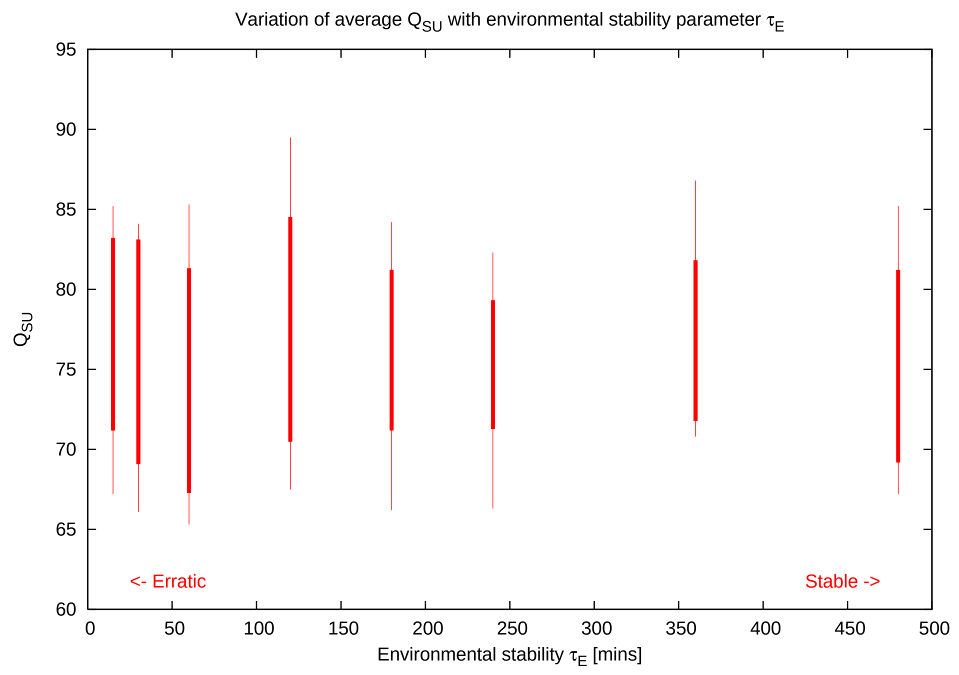


Figure 10.8: Variation of Q_{SU} with τ_E for selection model ζ_{FR}. Significant variation between runs, but little variation with τ_E.

10.5 Comparison of effect of varying look-ahead horizon parameter H for QLAS

A set of QLAS implementations with horizon lengths ranging from 0.5 to 6 hours were tested along with BDS yielding the results displayed in Fig. 10.10. The BDS examples yield Q_{SU} values averaging 82.9 with a variation ± 3.6. As can be seen, increasing H yields some overall improvement over the BDS though clearly this is not constant, i.e. some of the time BDS beats the QLAS examples, especially at lower τ_E as might be expected. Study of the amount of available execution time actually used (Fig. 10.5) shows somewhat unexpectedly that QLAS tends to be less efficient overall than BDS. This may be accounted for by the fact that BDS always tries to select something to do, QLAS will however sit idle for short periods in order to await a more lucrative observation. There are clearly some dangers in this approach, especially if the conditions should change during the wait thus leaving the awaited high-value observation no longer available, this then becomes wasted time.

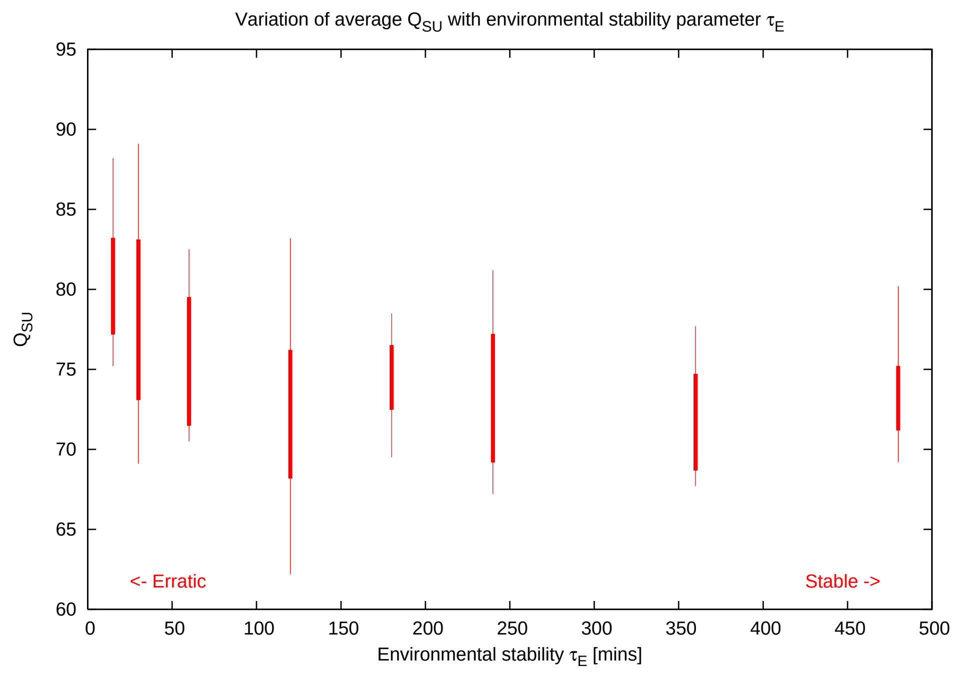


Figure 10.9: Variation of Q_{SU} with τ_E for selection model ζ_{RS}.Significant variation between runs. There appears to be a decrease in performance with increasing τ_E.

10.6 Comparison of effect of changing QLAS sequence count search parameter N_s

In order to examine the effect of varying the sequence count in QLAS, a series of simulations were run with N_s varying from 10 through to 10000 in logarithmic progression and with fixed values of $H = 1$ hour and the QLAS time quantum control parameter $\Delta\tau_q = 60$ secs. The experiments were performed under 3 different environmental scenarios with τ_E selected at 1,2 and 4 hours along with a baseline comparison performed using BDS. The results are given in Fig. 10.11 and show that the schedule quality improves significantly as we move to higher N_s values though with diminishing effect. At low N_s BDS always beats QLAS. These results are more or less as expected. The QLAS is looking for the best sequence from a potentially very large number of possible sequences. With small N_s it stands a good chance of missing the highest scoring sequences. As N_s increases we are searching more of the space of possible solutions.

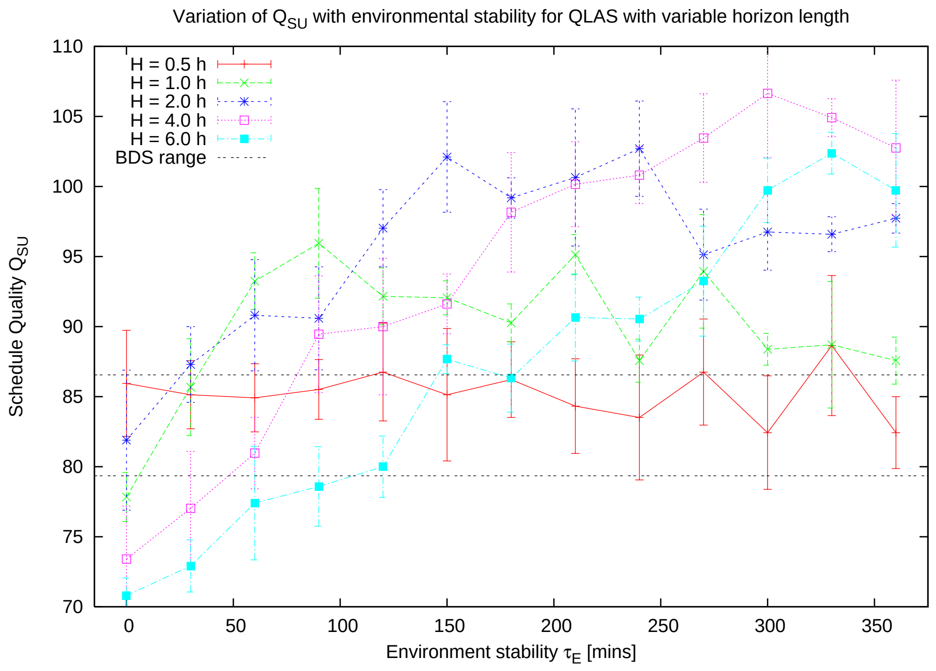


Figure 10.10: Effect of look-ahead horizon length H on Q_{SU} metric under a range of environment model stability parameter τ_E ranging from 0.5 to 6 hours. BDS provides fairly stable baseline with an average value of $Q_{SU} = 82.9$ and variation ± 3.6. QLAS shows reduced effectiveness (lower Q_{SU}) under unstable conditions (low τ_E) for all tested horizon lengths apart from $\tau_E = 0.5h$. As the environment becomes more stable (τ_E increasing) the longer horizon QLAS implementations become increasingly more effective.

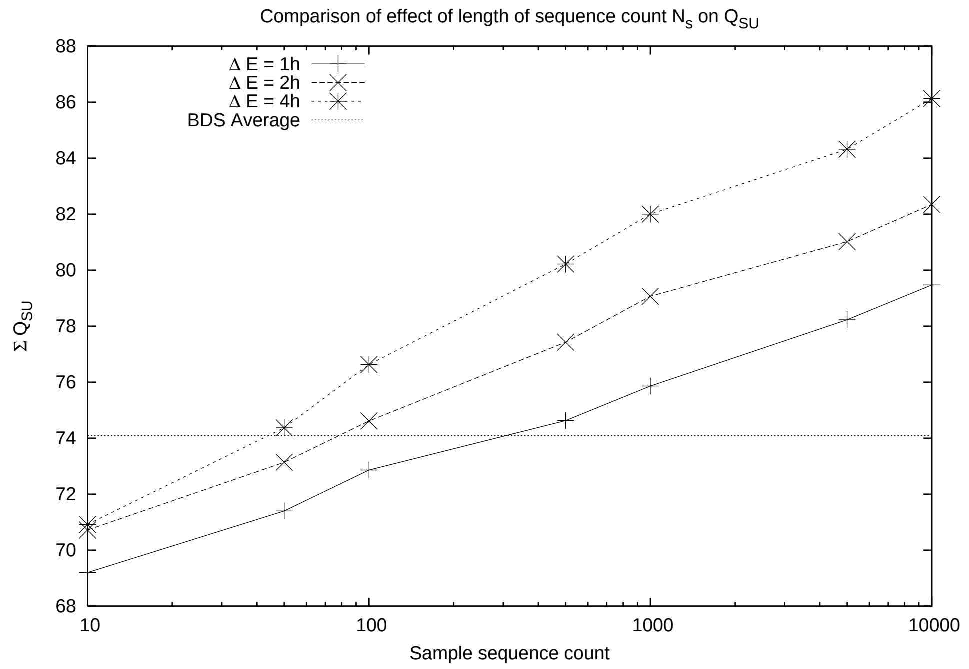


Figure 10.11: Effect of sequence count N_s on Q_{SU} under a range of environment model stability parameter ΔE. Q_{SU} is seen to increase with N_s in line with expectation as more of the search space of potential solutions is explored but with diminishing effect at high N_s.

10.7 Summary and conclusions

Using a QLAS scheduler it was found that as the degree of environmental stability increased we could profit significantly by employing longer look-ahead horizons. The rate of increase in the amount of improvement is seen to diminish with longer horizon lengths. It was found that by increasing the number of trials (a QLAS search control parameter) we could also achieve improvements, again however, the rate of increase was slow. This is a natural consequence of the random search mechanism employed, we should expect to see much faster improvements with a proper heuristic search mechanism in place. It was found that BDS could outperform many of the QLAS implementations when the environmental stability time τ_E was less than the look-ahead horizon, but that once $\tau_E \gtrsim H$ the LAS performance exceeds that of BDS.

11 Investigation into effects of disruptive events.

.During an observing night a variety of disruptive events can take place.

- Bad weather can stop observing for either an extended period or, when the weather variables are close to the upper and lower limits, for a series of short intervals, refered to as *spikes*.
- Mechanical, electrical and software problems can cause pauses in observing while recovery systems take over to effect repairs.

The lengths and frequencies of these interruptions are variable and have already been characterized (Sect. 4). In the case of a dispatch scheduler we might expect that little effect would be seen. Such schedulers are myopic, a feature generally regarded as non-optimal, though as we have already seen this can sometimes be advantageous. For the look-ahead schedulers which are investing effort to obtain the best overall reward by allocating appropriately to each time-slot, we might expect some potential degradation in performance.

A series of simulations were performed in order to investigate the effects of such disruptive events on scheduling performance.

11.1 Methodology

Simulations were performed using BDS and QLAS with horizons of length 0.5, 1, 2, 3, 4 and 6 hours. The environment model was set to fixed with good seeing. The Phase 2 model was populated with groups with small execution windows spread through the night. This means that the size of the pool remains more or less constant through the night but individual groups if missed are unlikely to be rescheduled later. The distribution of scores was arranged so that most groups are roughly level while a small fraction have distinctly larger scores. This is an attempt to *help* the QLAS to achieve higher scores than BDS as these are the sort of conditions where it might be expected to perform better and has in a sense *more to lose*.

In anticipation that the frequency of disruption rather than the total length of disruption is likely to have more effect, the simulations were performed with different numbers of disruptions but with the same total length (1 hour) of disruption time. A number N of disruptions each of the same length were scheduled to occur at random times during the night. We expect the total reward for an observing night to be reduced if a fraction of the night ϵ is *offline*. Consequently the summed reward for the *no-disruption* case ($N = 0$) is down-scaled by the factor $1 - \epsilon$. A total of 1000 simulations were performed for each combination of scheduler and number of disruptions over a single night of length 12 hours.

11.2 Results

The results of the simulations are shown in Fig. 11.1 with derived statistics in Table. B.21. For clarity, the line shown for BDS is an average of the results taken using the full range of *number of disruptive events*. The BDS results were fairly constant around the baseline value of ~ 118 with around ± 3.5% variation. For QLAS the situation is somewhat different. As can be seen, the effect of increasing numbers of short disruptions is to reduce the total reward. This effect is more noticeable for the longer horizon schedulers. At $n_e = 12$ QLAS with H=2 suffers a drop of 8.4% ± 5.6% whilst at $n_e = 0$ it performs better than BDS by 8.6% ± 4.3. We might infer that in these cases, because such a large investment has been made in obtaining the optimum sequence, any disruptions lead to higher instantaneous loss. We should however take note of the size of the error bars. These indicate the background level of variation between individual scheduling runs due to variations in execution time and ODB content. Overall the effect of disruption is quite small.

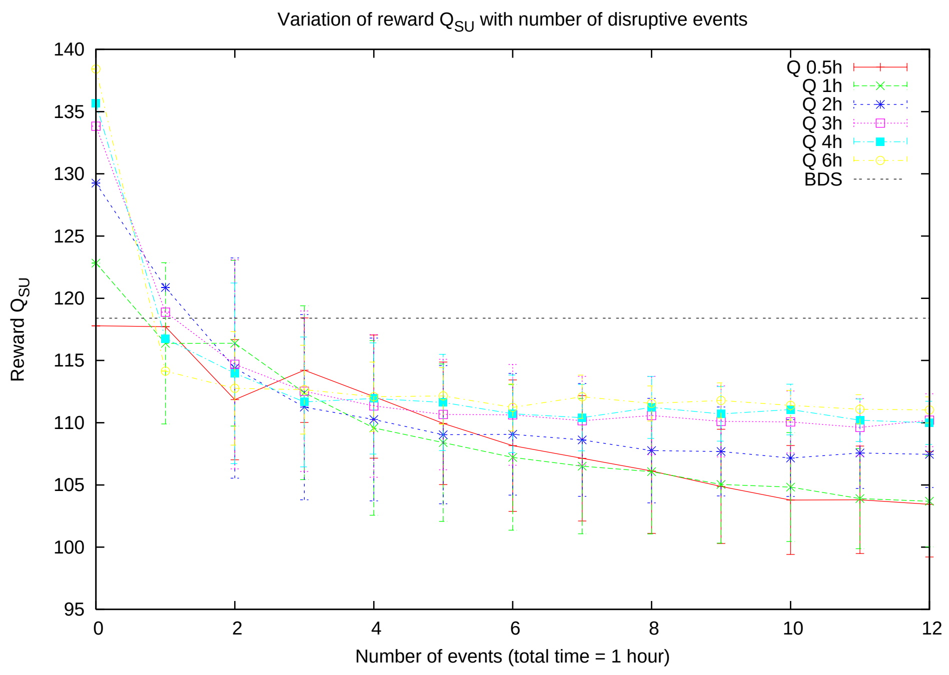


Figure 11.1: Effect of variation of number of disruptive events on schedule reward Q_{SU}. Results for $N = 0$ are scaled by a factor $1/(1-\epsilon)$ where ϵ is the fraction of night distrupted.

11.3 Summary and conclusions

Disruptive events, breaks in the natural execution cycle due to weather, mechanical or computer problems were investigated to determine their effect on schedule quality. It was found that for a given total disruption time, a large number of small disruption had a worse effect than a few long events. The effect was seen to be more pronounced for longer look-ahead horizons. The despatch schedulers were unaffected, the *myopic advantage*.

12 Investigation into effects ODB volatility

With the introduction at the start of Semester 09B of the *Web Start* based Phase 2 User Interface (P2UI) (Smith et al., 2010), users became able to add, delete and modify their observations at any time up to the moment they are scheduled. External software agents (Allan et al., 2006; Naylor et al., 2006) have since 2005 been able to enter new observations into the Phase 2 ODB at any time. Where these modifications occur during the night it seems reasonable to assume they will have some effect on the character and potential profitability of schedules which might be generated and introduce extra complexity into the scheduling operation.

The term *volatility* has been coined to describe the effect and the intention of this study is to determine the effect of this interaction on the various complexity and quality metrics.

12.1 Embedded instrumentation

As a first step and in order to characterize the scale of the problem, software has been embedded into the operational system to record these volatility events. A large number of potential agent interactions with the Phase 2 ODB are feasible, however many of these can be dismissed from consideration as they can be shown to have either little or no short-term potential effect on scheduling or occur very rarely. An instance of this would be a user adding a new group which does not start for several days time.

The principal effect of these significant events is to change the location and size of groups' feasibility windows. Where such a window is moved, the contention is reduced at those times contained in the *old* window, increased over the period of the *new* window but remains the same over any times where the *old* and *new* windows overlap. Where a feasibility window is reduced the overall demand will be increased. In addition, where a window is moved, the group's potential schedulability might be modified. This could occur for 2 reasons:- (*i*) The group's window might be moved to a time where the target is higher thus increasing its score, (*ii*) the window might be moved to a period of low contention among *low scoring* groups thus increasing its chance of selection.

It should also be noted that where contention is changed, quality metrics are also likely to be affected. E.g. where a *high valued* group is moved to a later time, the slot vacated may result in a lower valued group being selected at that time. The *moved* group may be shifted to a time where it is less valued than the other groups with which it is now in competition thus preventing it from being selected at all. The overall effect in this instance would be to reduce the schedule quality.

A qualitative study of the logs of volatility events in which the characteristics of groups before and after the event were compared, indicated that the following types of event were of most significance:-

Delete-Group Contention is reduced over any future feasibility windows of the group.

Update-Group Feasibility windows may be moved or their duration changed as discussed above.

Update-Sequence Execution time may change resulting in changes to feasibility windows.

In all the above cases, as each update event occurs, the embedded instrumentation records the time and type of event in a log and stores in serialized form the state of the group and its observation sequence before and after applying the changes. These can later be extracted for offline analysis.

12.2 Characterization of events

It is difficult to characterize the volatility of Phase 2 data with a single numeric value. However, it was found that volatility events could be characterized by a relatively small number of parameters.

time - t Simply the time the event occurs. Significantly we are only interested in events which occur during the observing night as these have the potential to affect a currently executing schedule. Events which occur during the day cannot affect schedules which have not yet been generated.

reach - ρ Represents the duration of the period of influence i.e. the range of times over which the event has an effect on the schedule metrics. The reach of an event can include several unconnected periods, e.g. for short-period monitoring groups there might be several windows of opportunity in a given night. The size, number and location of these windows may be changed by the event. Some four variations have been identified:-

- ρ_s **- Span reach** The total time between the first influence of the group either before or after the event and the last point of influence.
- ρ_t **- Total reach** Is defined as the total time during which some influence takes place either before or after the event or both.
- ρ_c **- Change reach** Represents the time during which the influence before and after the event is different i.e. greater or less but not the same.
- ρ_d **- Difference reach** Is similar to the above but the sign of the change is included.

proximity π Is defined as the interval between the event time and the start of the period of influence

magnitude The size of an event can be guaged by examining its effect on standard metrics. e.g. $\bar{\Delta_C}$ is the change to the average contention over the night introduced by the event.

Importantly, changes may have a variety of ranges - e.g. a change may be to add a group which starts in 2 days time. Another group may be added which can start in the next 5 minutes and needs doing in the next hour. One group may have a single execution, another adds an execution every 2 hours for the next month.

With reference to some standard complexity metrics for groups before and after an event with execution times x_a and x_b, we can easily derive the effect of an event as follows:

Change in average contention:

$\Delta C_c = \frac{\rho_a - \rho_b}{T}$

Change in average demand:

$\Delta C_d = \frac{1}{T}\left(\frac{\rho_a x_a}{x_a + \rho_a / n_a} - \frac{\rho_b x_b}{x_b + \rho_b / n_b}\right)$

12.3 Analysis of events

An extractor utility was developed to analyse information from the recorded logs. For each event the extractor first determines if the event time is of significance - events during the day are ignored. The state of the group before and after is determined and the old and new execution times are calculated. The total *reach* (ρ) and *proximity* (π) parameters are then determined. Events where π is greater than the length of the current remaining night are then discarded (they cannot affect tonight's schedule). The remaining events are potentially able to change the C and Q metrics for the night.

The excerpt below shows a series of events during a 1 minute period on 23 June 2010. The first excerpt shows the log for this period, we see that a total of 4 ADD_GROUP and 4 UPDATE_SEQ events occur. The ADD_GROUP events create a new group along with its various observing and timing constraints, however it is unpopulated - i.e. there is no specification of what to do. The subsequent UPDATE_SEQ event adds the neccessary observation specification.

```
2010-06-23 21:11:01 ADD_GROUP  p2update_201006233_29.dat
2010-06-23 21:11:01 UPDATE_SEQ p2update_201006238_30.dat   p2update_2010062310_31.dat
2010-06-23 21:11:01 ADD_GROUP  p2update_2010062331_32.dat
2010-06-23 21:11:01 UPDATE_SEQ p2update_2010062338_33.dat  p2update_2010062342_34.dat
2010-06-23 21:12:01 ADD_GROUP  p2update_20100623105_35.dat
2010-06-23 21:12:01 UPDATE_SEQ p2update_20100623111_36.dat p2update_20100623113_37.dat
2010-06-23 21:12:01 ADD_GROUP  p2update_20100623134_38.dat
2010-06-23 21:12:01 UPDATE_SEQ p2update_20100623140_39.dat p2update_20100623141_40.dat
```

In the second excerpt the extractor has paired up the above events and shows that 4 new observable groups have been added. The dashed segment after each text entry shows the feasibility window(s) of the new groups over a series of 10 minute intervals. A dash indicates no feasibility, a number indicates the fraction of a 10 minute period where the group is feasible.

```
DATA 2010-06-23 21:11:01 0 8580000  0.00 0.04 209  143 [--------------------0999999999999991-------------------------]

DATA 2010-06-23 21:11:01 0 14340000 0.00 0.03 159  239 [---------------0999999999999999999999997---------------------]

DATA 2010-06-23 21:12:01 0 15180000 0.00 0.03 164  253 [----------------59999999999999999999999996-------------------]

DATA 2010-06-23 21:12:01 0 16440000 0.00 0.02 147  274 [--------------29999999999999999999999999990------------------]
```

The final reduction of the data is shown in Table. 12.1 below. The example shows 4 new groups complete with observation sequences (this can be seen from the fact that the *before* and *after* values of demand (C_D) change from 0 (zero) to a real value. The groups are clearly of the same form (the execution times (X) are identical), this coupled with the rapidity of creation suggests they were generated by an external software agent

Time	$\mathbf{C_D}$ (Before)	$\mathbf{C_D}$ (After)	**X** (mins)	π (mins)	ρ (mins)
2010-06-23 21:11:01	0.0	0.04	6.6	209	143
2010-06-23 21:11:01	0.0	0.03	6.6	159	239
2010-06-23 21:12:01	0.0	0.03	6.6	164	253
2010-06-23 21:12:01	0.0	0.02	6.6	147	274

Table 12.1: Short extract of a section of the reduced volatility event table. The example shows 4 new groups complete with observation sequences.

Fig. 12.1 shows the distribution of the proximity measure (π) taken from the processed volatile update events. There are 2 peaks at 1 minute and 60 minutes accounting respectively for 47% and 23% of the events. These were found on detailed investigation to be due to automated inputs from an external agent for a microlensing program (Tsapras et al., 2009). The remaining events are mainly due to manual interaction via the Phase2 UI by users. In particular, analysis of the actual event sequences shows that typical user interactions make only small changes to the schedulability of groups during the night, typically just changes to the execution sequence which generally have little effect on the schedule. Most user interaction is during the daytime so does not affect the schedule in the coming night.

The distribution of the span (ρ_s) of the events is shown in Fig. 12.2 and shows peaks at 1 and 4 hours with a relatively smooth underlying distribution cutting off after around 5 hours. Both peaks were found to be due to external agents - the 1 hour peak is due to a series of groups with short proximity measure and 60 minute flexible observing period. The cutoff after 5 hours appears to be due to groups with longer periods of activity (typically 24 hours) but relatively low (galactic bulge) targets which set after a few hours thus cutting their observability window at that point.

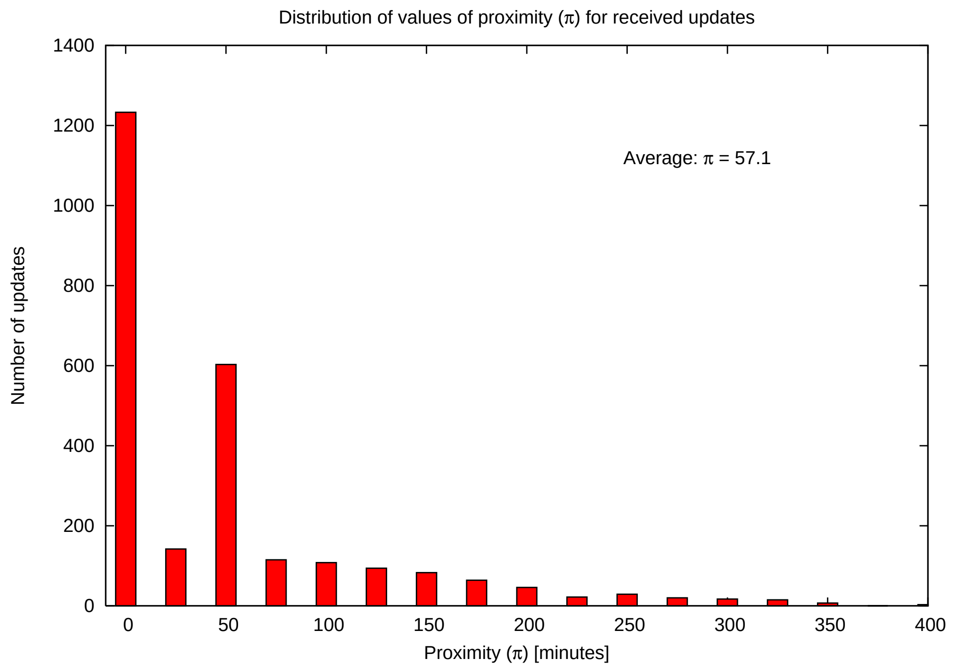


Figure 12.1: Distribution of proximity measure π for volatile updates received. Peaks occur at 1 minute (44%) and 60 minutes (23%) are accounted for by automated inputs from an external microlensing project's agent (Tsapras et al., 2009).

The recorded event data was processed to count the rate of arrival of events on a per-day basis from the start of the recording period (March 2010) until the end of the period (October 2010). A gap occurred from mid-March until end of April where the recording software was non-functional. From Fig. 12.3 we can see that the event rate peaks during June/July with rates of up to 80 events per day. This corresponds to the

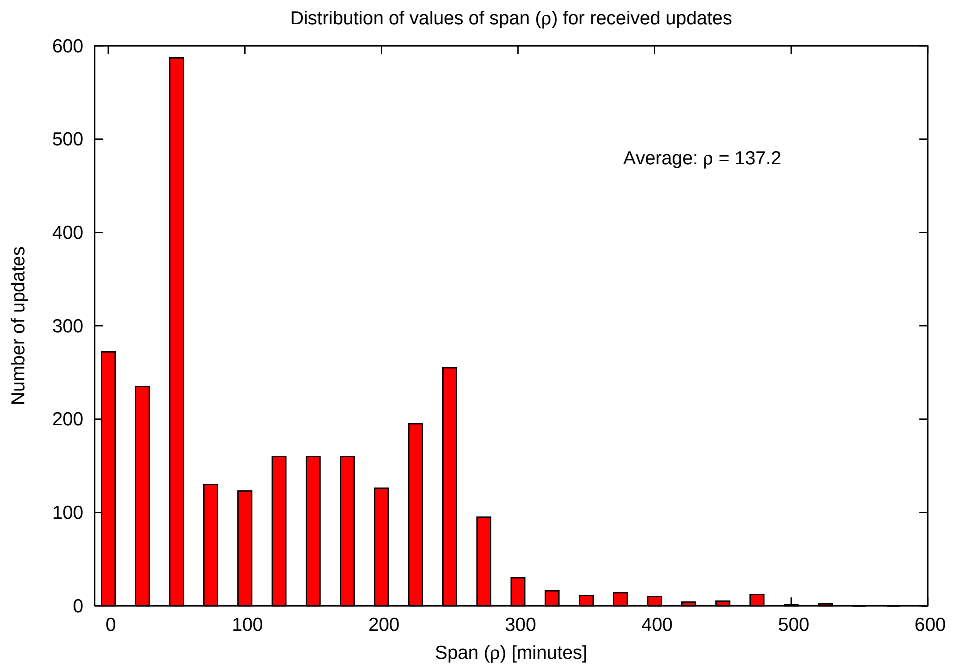


Figure 12.2: Distribution of span measure ρ for volatile updates received. There are peaks at 1 hour and 4 hours with cutoff above 5 hours. These are due to a series of automated agent inputs which start either almost immediately or with a 4 hour delay and have flexible timing constraints with 60 minute duration. The 5 hour cutoff is due to longer period flexible groups which set within their observing windows.

peak period of galactic bulge observing by the microlensing program. These events are found to consist exclusively of new observations and so add executable time to the schedule. Further analysis Fig. 12.4 shows that during this period up to 300 minutes (5 hours) of potential observing may be added per night - a significant fraction of the observing night. Unfortunately data showing the total amount of observing available on these nights is not easily obtainable so it is difficult to determine the overall effect on schedules. It is also found from examination of schedule logs that on occasions though many observations are added these will often be in competition with each other for observing time and so cannot in fact all be executed in practice.

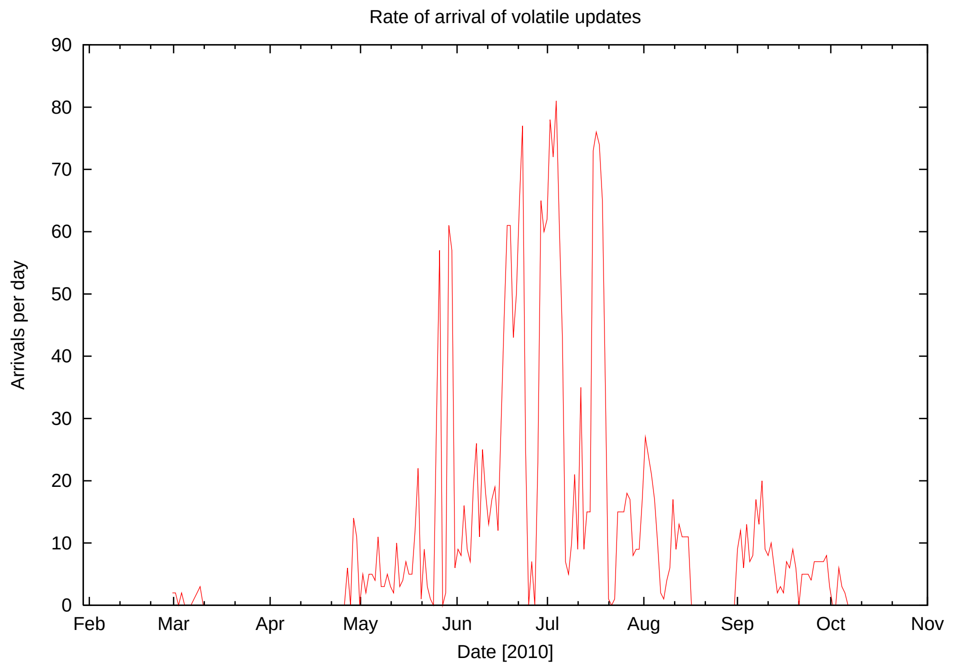


Figure 12.3: Rate of arrival of volatile updates. Peak rate of arrivals occurs June-July corresponding to the peak period of galactic bulge observing by the microlensing program with up to 80 events per day.

12.4 Simulation

In order to quantify the effects of volatility it was decided to perform a series of simulations under varying conditions of volatility. The mechanism chosen to implement is the addition to the simulation architecture of a Volatility Scenario Generator (VSG),

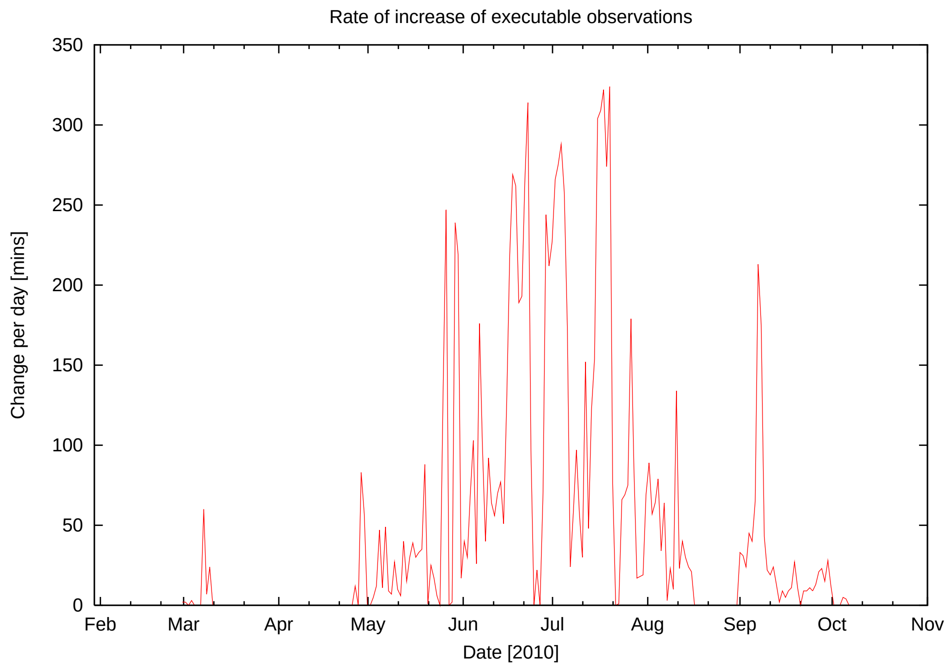


Figure 12.4: Rate of increase of executable observations. During the peak period, up to 300 minutes of new observations arrive per day amounting to around 50% of the potential observing load.

henceforth denoted by the symbol $\mathcal{V}$.

A simple volatility generator was devised based on a specified average rate of arrival (λ). When the simulator calls on the VSG to generate events over a specified interval (see Fig. 6.2 and Sect. 6.2) the decision on how many events (k) to generate is calculated based on the poisson distribution $P(k) = \frac{\lambda^k e^{-\lambda}}{k!}$. A random number generator generates a number $r \in [0, 1]$. The smallest number k such that $r > P(k)$ is the number of events to be injected.

12.5 Investigation into the effect of proximity (π) on schedule quality

In this experiment the effect of varying the proximity of groups added via volatility events was investigated. The scoring model was modified to simulate a typical scoring sequence during an observing night. The base groups (those already available via the

Phase2 model) produce scores randomly within bounds $[V_{lo}, V_{hi}]$. The groups added via the VSG are specially tagged and have scores set at a predetermined level V^* with a small amount of noise added.

A total of 7 scheduler models were tested; BDS - a basic despatch scheduler, QLAS - a look-ahead scheduler with horizons of 1, 2 and 4 hours and ELAS - the enhanced look-ahead scheduler with the same three horizons as QLAS (Sect. 6.4). The environment model was set to a fixed state (good seeing and photometric) for all runs. Simulations were performed as follows:-

- A set of preliminary simulation runs were performed with each of the selected schedulers but with no volatile events in order to give a baseline.
- Simulations were then performed with π varying logarithmically between 1 minute and 6 hours in accordance with the limits found from recorded volatility events (Sect. 12.3). For each π value ρ was chosen from a random distribution in the range $[1, 120]$ minutes. For each π value 100 simulations were performed with each of the selected schedulers. The value of V^* is set close to V_{hi} so tagged groups are generally favoured relative to base groups.
- A third set of simulations were performed with each scheduler in which all of the volatile events were instead fed in at the start before any scheduling took place to allow an upper bound to be set on the potential reward.

12.6 Results

The baseline measurements are shown in Table. 12.2. Q_0 is the mean value of Q_{SU} measured on simulations with no volatile events. The column labelled Q_{HI} shows the mean results for simulations with *all* the volatile events injected in advance while ΔQ is the difference between these and represents the parameter used to scale the y-axis in the following graphs. As can be seen, in the case of BDS there is a significant improvement when the volatile events are added in. The figures in parenthesis are σ values for these means.

Results for the QLAS and ELAS schedulers are displayed in Figs. 12.5 and 12.6. The quantity plotted on the y-axis is the improvement over the baseline Q_0 divided by ΔQ The results for the various QLAS horizons all show improvement over the baseline values. Most noticably the *rise time* appears to correlate with the horizon length. QLAS(1) picks up faster than QLAS(2) and QLAS(4). The suggestion here is that the QLAS does not see the changes which occur in a time short compared to its horizon and thus cannot react to them. In fact the QLAS is effectively unavailable for scheduling during the execution of a horizon length sequence, any events occuring in this period are not seen until the end of the execution. However because the events are not synchronized to occur during this period, the QLAS may in fact be at any stage in a sequence's execution so will see some of the tagged groups sooner than 1 full horizon length.

Figures 12.7 through 12.10 show comparisons of the schedulers QLAS and ELAS of the same horizon lengths. Associated statistics are given in Tables. B.22 through B.25. It is clear that ELAS performs better than QLAS especially at low values of proximity (π) and that this improvement is greater for schedulers with longer horizons lengths.

Model	$\mathbf{Q_0}$	$\mathbf{Q_{HI}}$	$\mathbf{\Delta Q}$
B	96 (3)	106 (3)	10
Q_1	102 (2)	113 (4)	11
Q_2	105 (4)	117 (4)	12
Q_4	108 (4)	119 (3)	11
E_1	101.2 (3)	114.3 (4)	13
E_2	105.3 (4)	116.5 (4)	11
E_4	107.8 (4)	119.3 (3)	11

Table 12.2: Mean values of Q_{SU} measured using the baseline simulations. Q_0 is the mean value of Q_{SU} measured on simulations with no volatile events. Q_{HI} indicates the mean value of Q_{SU} measued on simulations with *all* volatile events injected at the start. ΔQ is the difference and is the parameter used to scale the y-axis in Figs. 12.5-12.10 .

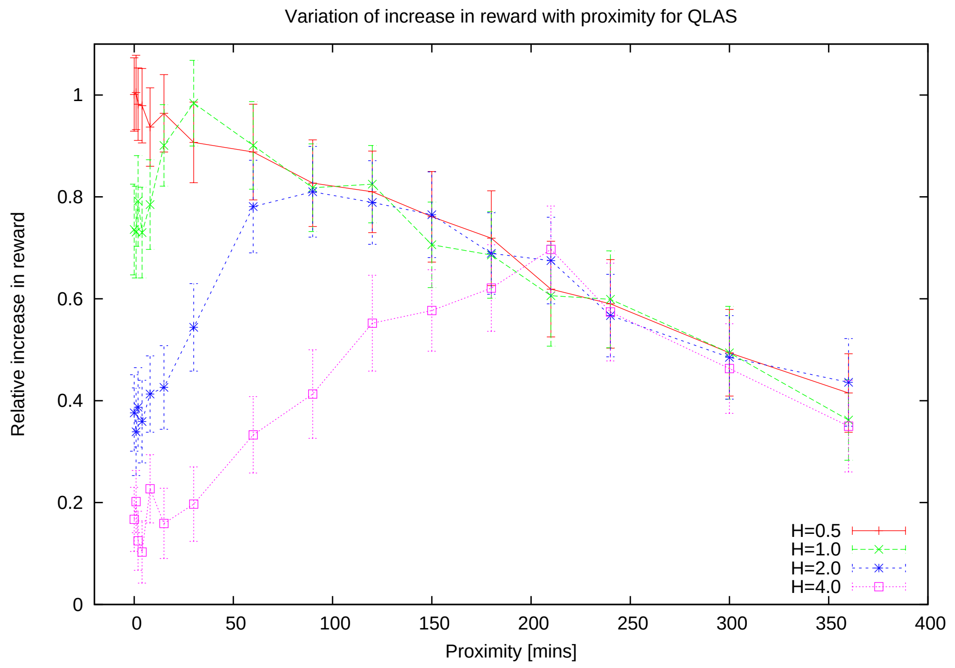


Figure 12.5: Effect of proximity (π) of volatile events on schedule quality (Q_{SU}) for QLAS. All QLAS horizons show improvement over the baseline values. Most noticably the *rise time* appears to correlate with the horizon length. QLAS(1) picks up faster than QLAS(2) and QLAS(4). The suggestion here is that the QLAS does not see the changes which occur in a time short compared to its horizon and thus cannot react to them.

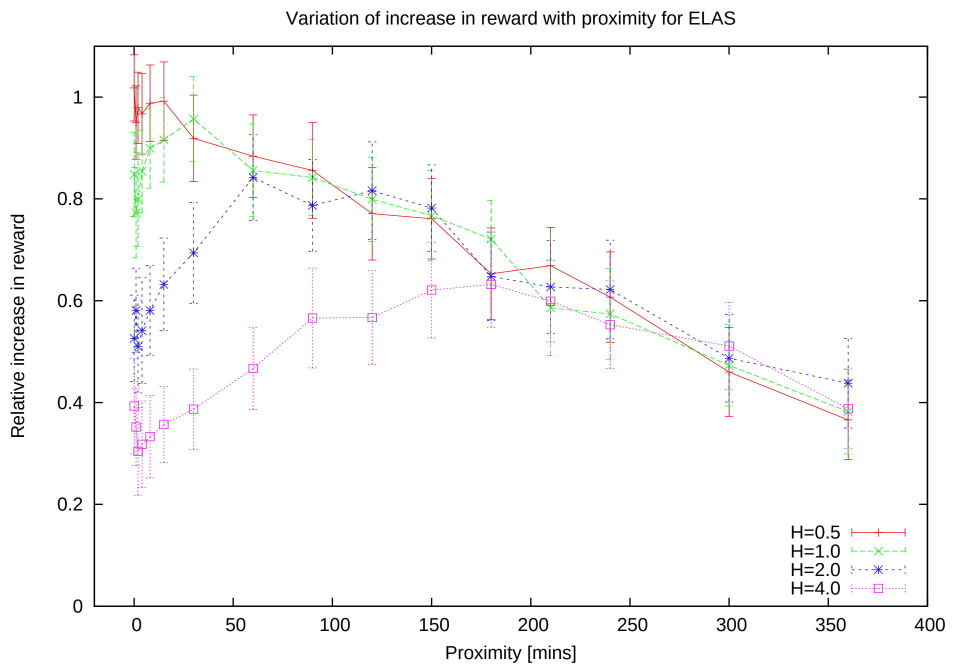


Figure 12.6: Effect of proximity (π) of volatile events on schedule quality (Q_{SU}) for ELAS. The ELAS schedulers follow a similar pattern to QLAS but the rise-time is slightly shorter, i.e. the ELAS schedulers react faster to volatile events.

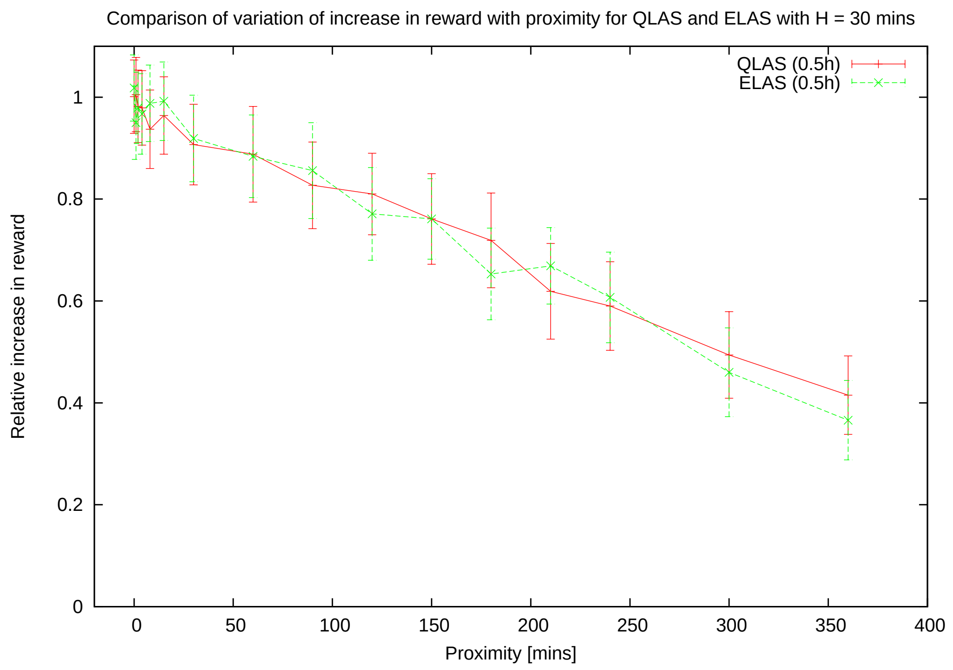


Figure 12.7: Comparison of effect of proximity (π) of volatile events on schedule quality (Q_{SU}) for QLAS and ELAS with $H = 0.5h$.

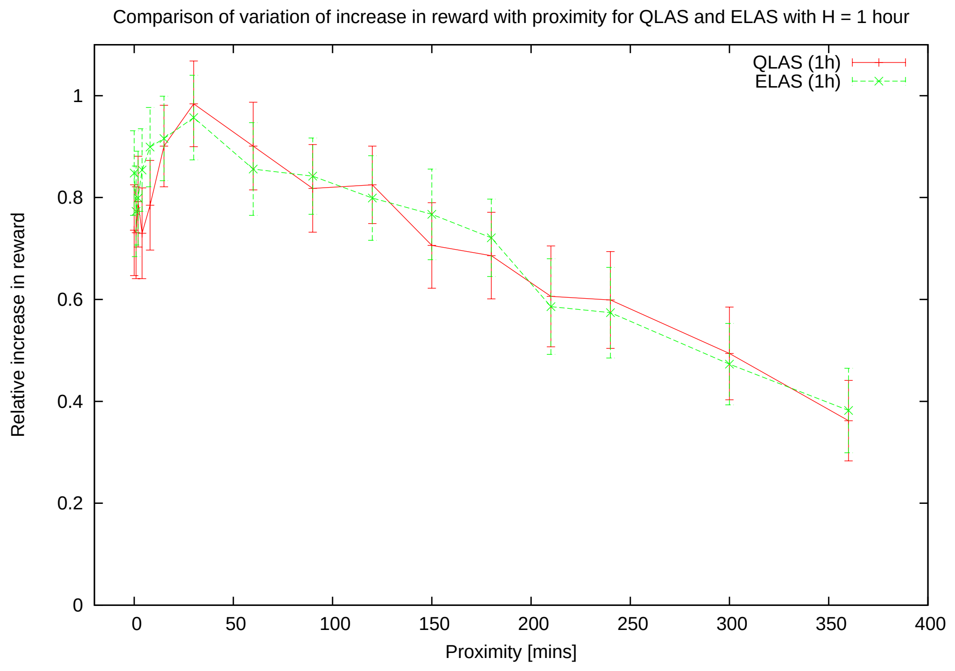


Figure 12.8: Comparison of effect of proximity (π) of volatile events on schedule quality (Q_{SU}) for QLAS and ELAS with $H = 1h$.

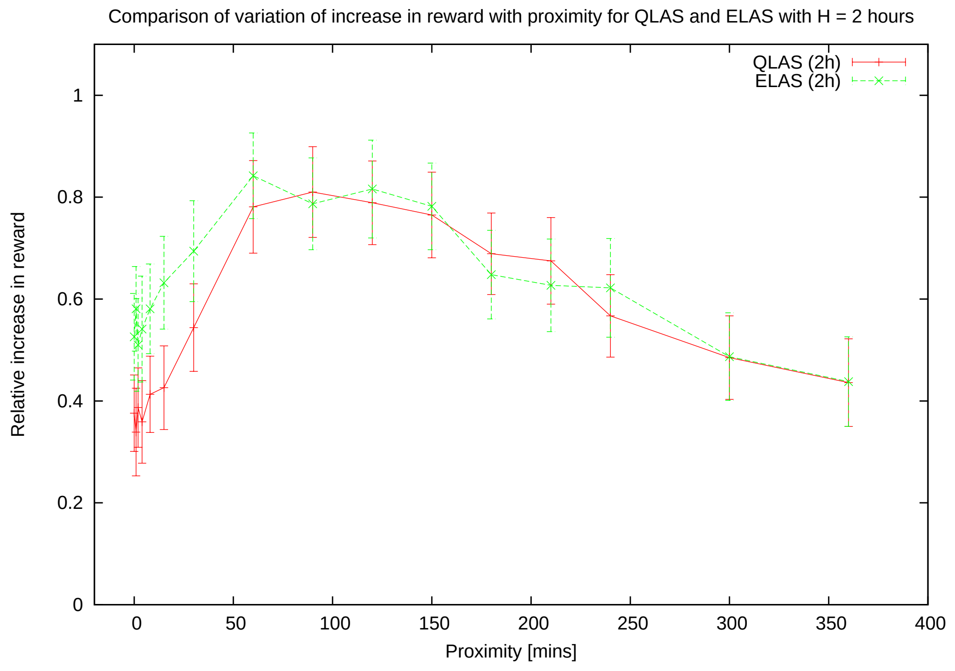


Figure 12.9: Comparison of effect of proximity (π) of volatile events on schedule quality (Q_{SU}) for QLAS and ELAS with $H = 2h$. This plot shows some improvement in quality by ELAS over QLAS at lower proximity.

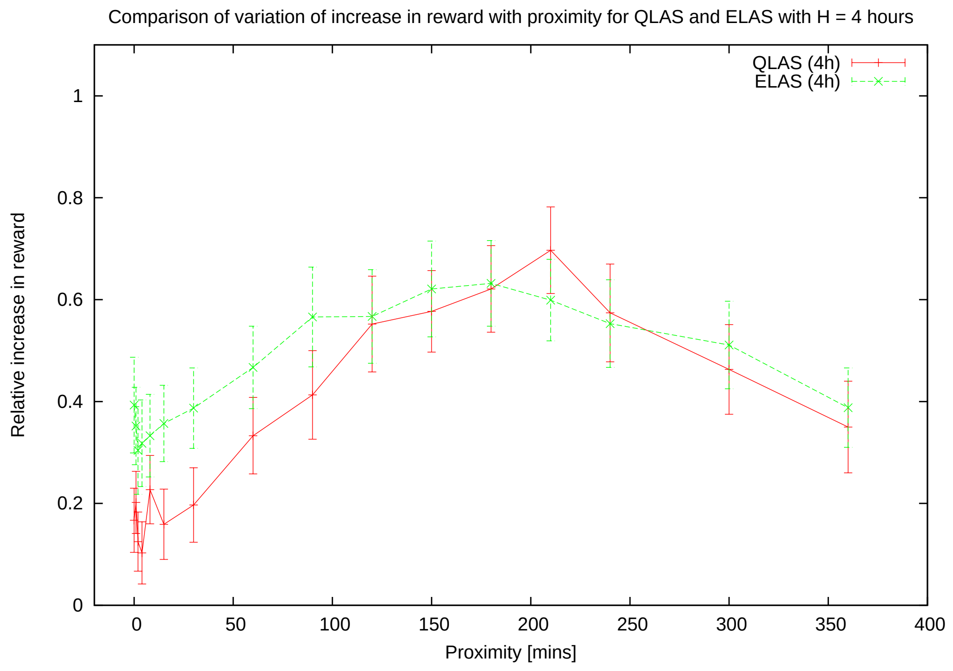


Figure 12.10: Comparison of effect of proximity (π) of volatile events on schedule quality (Q_{SU}) for QLAS and ELAS with $H = 4h$. This plot shows the most improvement in quality by ELAS over QLAS at low proximity.

12.7 Summary and conclusions

Embedded instrumentation was used to record changes to the Phase 2 ODB, termed volatile events. Some simple metrics were designed with which to characterize these events. It was found that at busy times up to 50% of the nights potential observations can arrive after the start of night's observing.

In simulation experiments it was found that for any given look-ahead horizon H there was an optimum value of proximity (one of the characterization metrics) at which the highest relative improvement occured. This was found to be related to H. An enhanced look-ahead scheduler, ELAS which included a sequestration policy was introduced and this was seen to yield further improvement in schedule quality. The effectivenss of ELAS relative to QLAS was seen to increase with increasing horizon length.

13 Investigation into effect of reliability of stability prediction

13.1 Introduction

As has already been seen in Sect. 10, the choice of length of look-ahead horizon is dependant on the stability of environmental conditions. This experiment is designed to examine what happens if the actual stability is better or worse than that predicted. In such cases the chosen horizon length will be an under-estimate or over-estimate. If H is an under-estimate then we are wasting some of the potential advantage to be had from a long period of stability. If H turns out to be an over-estimate then we might find that the sequence is cut short and we lose some of the groups which had been lined up.

13.2 Experimental setup

A simulation was setup using a Phase 2 model containing 500 flexible groups with feasibility windows with lengths from 30 to 120 minutes spread throughout a single night. The model was seeded with 50 groups with scores varying strongly with time over very short (15 minute) feasibility windows. These groups could easily be missed by a despatcher and were intended to boost the performance of look-ahead. A fixed execution timing model was chosen to minimize extraneous random variations.

Using a set of 8 reliability factors q chosen from the range 0.1 to 2.0, and 5 look-ahead horizons H chosen from the range 0.5 to 6 hours the stability parameter of the environmental model was selected as $\tau_E = H/q$. Where $q < 1.0$ this represents an under-estimate while $q > 1.0$ represents an over-estimate of environmental stability. For each combination of H and q, 100 simulations were performed with different environmental scenarios generated by the environment model over the selected night and value of Q_{SU} measured. A seperate set of simulations were performed with τ_E set equal to the value of H and are used as a baseline.

13.3 Results

The results of these simulations are plotted in Fig. 13.1. The quantity plotted against the y-axis is the ratio Q_{SU}/Q_0, with Q_0 being the baseline value of Q_{SU} with $\tau_E = H$. The error bars represent variation due to different environmental scenarios on each run. The graphs show a peak value in each case with q around the value of 0.5 to 0.8. Runs with horizon of 0.5 hours show the least variation and no peak. Other horizons show a rapid rise to the peak value then a more gradual fall-off, levelling of to around 0.9 to 0.95 of the baseline value. It is notable that the peak values occur when $q < 1$ which is most likely a feature of the environmental scenario models, the mean and median of the stable period distribution are not equal, with the median being less than τ_E.

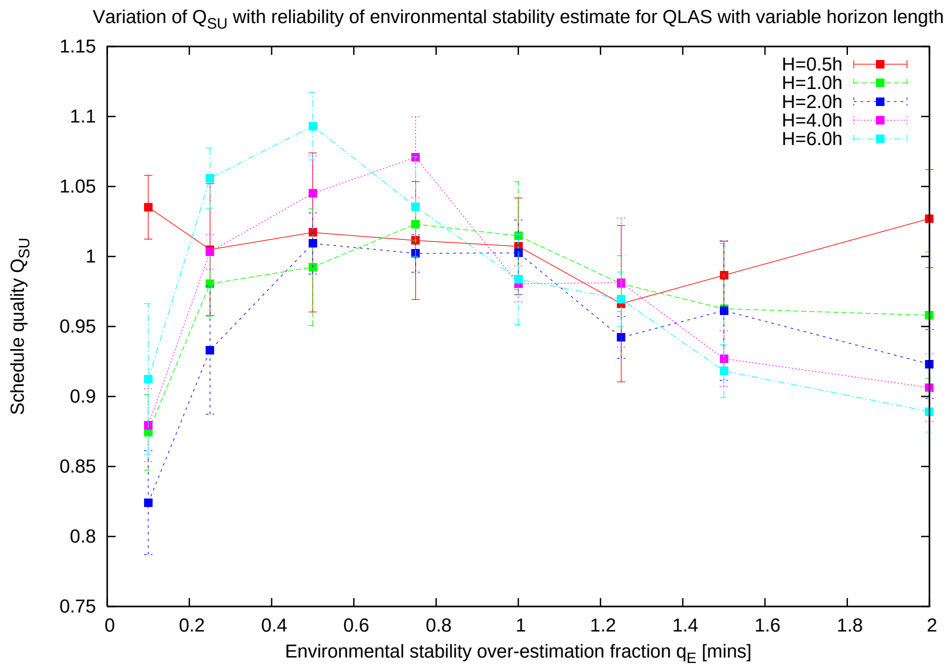


Figure 13.1: Variation of qsu with q for different horizon lengths. For $H = 0.5$h there is little effect, whilst longer horizons appear to rise to a peak at $q \sim 0.8$ then a gradual decay, levelling of to around $0.9Q_0$.

13.4 Summary and conclusions

The effects resulting from the accuracy with which we are able to predict the environmental stability timescale was investigated. It was found that when determining the horizon length H, if we either over or under-estimate the duration τ_E of stable conditions we get poorer results. The optimum situation appears to be when $H \simeq f \times \tau_E$ where f is around 0.7 to 0.8.

14 Conclusions

14.1 Metrics

In Sect. 3 a number of Complexity (C) and Quality (Q) metrics were considered. Demand C_D and contention C_C were suggested as useful measures of the complexity of a scheduling problem. Later in Sect. 9.4 it was shown that contention can also provide an indication of the potential utility achievable by a scheduler, in that complex (high C_C and C_D) ODBs tend to give higher overall quality measures. Basically there are likely to be a higher population of quality observations to choose from and so more chance these will be selected than in a low contention ODB, where at times low quality observations might be chosen for lack of anything better.

A general utility measure combining user and enterprise measures of quality was suggested (Eq. 7) and provides a potentially interesting area for further study.

For the current thesis a set of simpler (primitive) quality measures were designed and used in the simulation experiments. Some associations were suggested between these quality metrics and the accepted scheduling and planning timescales. Of the available utility measures, the two most promising for future use in longer term planning are the yield tracking (Q_{YT}) and target demand (Q_{TD}) metrics. They are also the most challenging and time-consuming to calculate and for this reason were not used seriously other than in the early shakedown experiments as proof of concept.

14.2 Characterization of operating environment

A detailed study of the operating environment was made, gathering information from a variety of sources with a view to employing this information in developing future longer horizon scheduling and planning systems.

Meteorological information was obtained from logs of the telescope's own weather monitoring system (WMS) and from nightly observing logs. In terms of the telescope, weather is classed as good or bad depending on whether a specific set of conditions are satisfied. The weather on site was found to be good for about 80% of the time averaged over the year. There is, perhaps not surprisingly, generally more good weather in summer than winter. Typically it was found that there is $< 5\%$ bad weather in June rising to as much as 60% in February.

The major contribution to bad weather is high humidity accounting for around 18% of all time (90% of bad weather). Other factors such as high winds and freezing temperatures account for only a few percent of the bad weather.

Using an analysis of the distribution of lengths of good and bad periods, a simple prediction model was designed in which it was assumed the weather would continue in its current condition for a period similar in length to the amount of time which it has already been in that state but with an exponentially decaying likelyhood of changing to the alternate state. This model was tested against recorded data and was found capable of anticipating the length of the current period of weather with a degree of accuracy exceeding the long-term average for up to 30 hours ahead though with decreasing accuracy. Further analysis of the WMS data shows that continuous runs of bad weather (days on which the amount of time classified as bad exceeds a specified threshold) indicate that for a threshold of 90%, around 50% of bad weather runs are < 5 days long with $< 10\%$ of runs exceeding 15 days. This information could be useful in longer-horizon planning, for instance to determine whether a particular group might be observable in the next few nights or whether it would be safer to observe it tonight.

In terms of combining the various sources of weather information to influence decisions over various planning and scheduling timescales, the following assignments are suggested:-

- Operational planning. As we are only interested in the weather a few hours ahead in order to determine the sequence order of observing, the length statistics derived in Sect. 4.3.3 would be appropriate. A typical question to answer might be:- *What are the chances I can do group X in 3 hours as I will get a better quality observation then compared to executing it now?*.

- Tactical planning. The type of decisions to be made at this level include answering questions like:- *What are the chances of performing X in the next 3 days - I can do X on any of those nights with little difference in reward but if I do X tonight I miss my chance of doing Y?*. External forecast information, ideally from automated sources, or fed in by operations staff on a night by night basis might be used to predict weather downtime for tonight and the next few nights up to the limits of accuracy of the forecast.

- Strategic planning. From this perspective we are asking question like. *What are the predicted effects on data yield over the next N nights on group Z based on the likelihood of execution on those nights?*. Climatological information might be used to predict general probabilities weeks and months ahead.

Details of the atmospheric seeing, measured by a real-time pipeline operating on data from the telescope's main imaging camera were extracted from the data archive and reduced taking into account pixel-scale, binning, filter wavelength and target elevation above the horizon. The results indicated that the median R-band seeing achieved is around 0.95". It was also noted that median seeing is best in summer, typically around 0.78" with average of 1.0" and poorest in December with a median value around 0.93" and average over the winter months of 1.5". It was further found that typically, and contrary to popular belief, no systematic variation in seeing quality occurs over the course of the night.

Using technical downtime information from the nightly operations logs gathered over a period of the first 3 years of operation of the telescope it was found that most nights (> 72%) have less than 5% downtime while some 8% of nights suffered nearly 100% downtime. In terms of use in prediction, there was no readily discernable pattern to technical downtime.

14.3 Architecture

The original brief was to develop an architecture with which to build a variety of schedulers along with a simulation framework with which to characterize and test schedulers on a variety of scheduling scenarios. Initial work on the LT scheduler deployed at the start of robotic operations provided clues as to the range of components which would be required.

A comprehensive architecture was then designed taking into account these considerations. The architecture contains a large number of pluggable interface components which allow numerous scheduling paradigms to be created. In the subsequent experiments two main paradigms were tested. BDS is a simple despatch scheduler while QLAS is a look-ahead scheduler with a pre-designated horizon size.

Using the simulation framework, the pluggable nature of the architecture allowed a number of experimental contollers to be quite straightforwardly setup and used to test the schedulers. The number of components available means that for anyone else using the framework there would be something of a learning curve required to use it, however much of the setup is fairly standard and it has proved easy to simply use an existing simulation controller as a template, then by making small changes to just one or two models adapt it to perform alternative experiments. Often just the Phase 2 model needs changing to create a different population distribution or to vary the seeding of high-valued groups.

Much of the information gained in developing the framework leads me to conclude that it would be quite feasible to extend the use of this architecture to other projects, perhaps with radically different Phase 2 concepts despite the critical dependance of much of the architecture on the very nature of the adopted Phase 2 model. In such a case it would be neccessary for either the project to adapt its Phase 2 concept to match the one used in this thesis or to adapt the architecture to an alternative Phase 2 model.

14.4 Experiments

Man against Machine It was shown that a human scheduler using rules which may not be known to the scheduler himself is capable of generating effective, high-quality schedules as measured by various Q metrics. In some cases the human scheduler was able to outperform an automated (BDS) scheduler on certain metrics. By studying the variation of quality metrics against those of the automated scheduler it was possible to deduce the approximate relative weighting factors used by the human scheduler as being given approximately by $0.8f_{el} + 0.2f_{pr} + \epsilon f_{rn}$ where ϵ is very small ($\ll 0.1$).

Scoring This series of investigations showed that the effect of changing the weights applied to the various heuristic metrics used in making scheduling decisions have little effect on each other. When we change any given scoring (f) metric (sect. 5.7.1) the corresponding quality (Q) metric (Sect. 3.3) is affected though not as much as might have been anticipated. There is generally little effect on any other Q metric. The choice of weighting determines the character of the schedule but only the dominant f metric has any real effect while the other metrics appears like noise and suppress the effectiveness of the chosen f metric. The overall conclusion that can be taken from this is that whatever qualities we want from a schedule must be very actively selected for in the scoring and selection models.

Complexity In order to compare the capabilities of generated Phase 2 models with those from real ODB snapshots a series of experiments was performed. These experiments also investigated the effect of database (Phase 2) complexity or *weight*, measured using the average contention measure C_C on scheduling. It was found that by suitable adjustment of the generator parameters, models could be created with similar characteristics to real ODB snapshots. There was however significant variation in the qualities of these models. The scheduling experiments showed that at low levels of complexity there is considerable variation in the fullness (Q_{XT}) of schedules generated by all scheduling paradigms. As the level of complexity increases the degree of variation decreases. The variation in quality (Q_{SU}) was found to be fairly constant at all levels of complexity but the average value was seen to increase with complexity for all scheduling paradigms whether despatch or look-ahead. For a despatcher the quality increased by around 17% between $C_C = 1$ and $C_C = 25$. For a look-ahead scheduler with $H = 4$ hours the improvement was up to 23%. It was also found that as the look-ahead horizon increases there is a general (though small) increase in quality with complexity.

Stability Using a QLAS scheduler it was found that as the degree of environmental stability increased we could profit significantly by employing longer look-ahead horizons. The rate of increase in the amount of improvement is seen to diminish with longer horizon lengths from a gain of 23% ± 8% for a 4 hour horizon down to a gain of 4.8% ± 8% for a 30 minute horizon. It was also found that by increasing the number of trials N_S (a QLAS search control parameter) we could also achieve improvements, again however, the rate of increase was slow. At $N_s = 100$ a gain of 4% over BDS was found, while at $N_s = 10000$ this increased to 12%. This is a natural consequence of the random search mechanism employed, we should expect to see much faster improvements with a proper heuristic search mechanism in place. It was found that BDS could outperform any of the QLAS implementations when the environmental stability time τ_E was less than the look-ahead horizon, but that once $\tau_E \gtrsim H$ the LAS performance exceeds that of BDS.

Volatility Embedded instrumentation was used to record changes to the Phase 2 ODB, termed volatile events. Some simple metrics were designed with which to characterize these events. It was found that at busy times up to 50% of the nights potential observations can arrive after the start of night's observing.

In simulation experiments it was found that for any given look-ahead horizon H there was an optimum value of proximity or lead-time (one of the characterization metrics) at which the highest relative improvement occured. This was found to be related to the look-ahead horizon H. Though QLAS yielded improvements over BDS of around 12%, this gain could drop by up to 40% of its value when the volatile events had lead-times away from this optimal value. An enhanced look-ahead scheduler, ELAS which included a sequestration policy was introduced and this was seen to yield further improvement in schedule quality by mitigating the *proximity-effect*. The effectivenss of ELAS relative to QLAS was seen to increase with increasing horizon length.

Disruption Disruptive events, breaks in the natural execution cycle due to weather, mechanical or computer problems were investigated to determine their effect on schedule quality. It was found that for a given total disruption time, a large number of small disruptions had a worse effect than a few long events. The effect was seen to be more pronounced for longer look-ahead horizons. For the look-ahead scheduler, a loss of up to 11% was experienced relative to the despatcher and up to 22% relative to its own best performance. The despatch schedulers were unaffected, the *myopic advantage*.

Prediction accuracy The effects resulting from the accuracy with which we are able to predict the environmental stability timescale was investigated. It was found that when determining the look-ahead horizon length H, if we either over or under-estimate the duration τ_E of stable conditions we get poorer results. The optimum situation appears to be when $H \simeq f \times \tau_E$ with f around 0.7 to 0.8. When the stability length is severely under-estimated a loss of between 5% and 17% in quality was found to occur. Where the stability length is over-estimated by a factor 2, a loss of 5% to 10% can occur.

14.5 Summary

A principal aim of this thesis was to develop the tools with which to build an adaptive scheduler and to determine under what conditions such adaptive behaviour would be neccessary. We have established that the two scheduling paradigms investigated do indeed behave differently with respect to operating conditions and that in particular the *stability* or *smoothness* of the operating environment is a critical factor in determining both which type of scheduler to employ and how it should be configured.

BDS is a *myopic* scheduler, it can only see what it is doing at the time and knows nothing about what future decisions it might need to make. This can be both an advantage and disadvantage. Under rapidly changing conditions, this myopism is a definite advantage. A look-ahead scheduler which is committed to its decisions for a period may suffer because either the conditions improve and it has already chosen to perform low quality observations, or the conditions deteriorate and the previously committed high quality observations become non-viable. The effects of large numbers of disruptions and volatile events have a similar effect in that future commitments may have to be broken to accomodate these effects, thus reducing the look-ahead advantage. BDS on the other hand does not see any of this instability and is able to respond instantly to changes.

Under stable conditions, we have seen that provided we are able to estimate the degree of stability accurately, the choice of a long look-ahead horizon gives considerably better results (up to 26% increase in Q_{SU}) than simple despatching.

A design for the adaptive scheduler is now suggested. It is clear that we need a way of accurately predicting these variables ahead for at least the length of the longest horizon we might consider. Once we have this information we can choose a look-ahead horizon to be roughly the length of the shortest stability period. However, when conditions are very *rough* in any of the operating parameters we should revert to despatch scheduling. The work of Sect. 12.3 also informs us that both an effective break policy and sequestration policy will help us to mitigate against instability so a scheduler like ELAS is more useful than QLAS.

The look-ahead schedulers tested used a very inefficient search algorithm. A BDS run on the deployed system takes around 1-2 seconds to perform feasibility tests on a typical pool of 500 groups. For a run of QLAS with a 2 hour horizon and 2 minute quantum there will be 60 slots to fill. If around 5-20% of groups are feasible at any time we might expect each quantum to test around 5-10 groups for feasibility. Therefore 5x60=300 tests per horizon taking around 0.6s. We need to run at least 100 and preferably more than 1000 sweeps to find a reasonable schedule so therefore 60 to 600 sec per horizon. Though perfectly accceptable for simulation experiments where time was not an issue, this would need to be improved upon significantly for an operational system.

It is clear from Sect. 3 that there is no *magic* quality metric which can be used to determine the optimum schedule. The decision on which metrics to use are very much dependant on operational and management preferences. The combined utility metric (Eq. 7) discussed in Sect. 3.3 incorporating both enterprise and user preferences is suggested as the way ahead.

It seems reasonable to suggest that a degree of forward planning might be useful. I have already alluded (Sect. 3.5) to the yield-tracking and urgency metrics (Q_{YT} and Q_{RN}) and how these might be useful in this context. Fig. 14.1 shows a possible way in which longer-term planning could be integrated. The higher planning layers use long-term quality metrics (fed through statistics from lower levels) to assess whether the scoring metrics and weights at those lower levels should be varied.

The work in Sect. 4.3.3 on forward prediction of weather condtions and run-length distributions could be useful in this context. An example might be where a group of high quality might be observed on several of the next few nights. However, it would be better to observe in 2 nights time if possible - perhaps the moon will be further from the target or the seeing is not especially good tonight. In such a case we would need to balance the likelihood of conditions improving over the next few nights against the possibility of the weather deteriorating to the extent that the group cannot actually be observed on those nights. Such effects could for example be taken into account using a scoring mechanism including a discounted future reward term (Eq. 18).

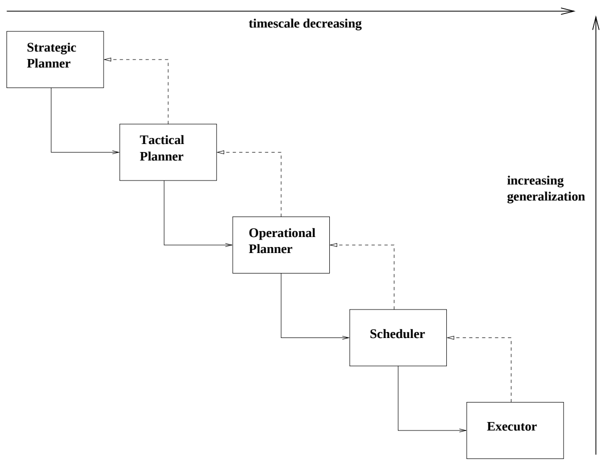


Figure 14.1: Interaction between planning and scheduling. Lower levels feed metric information to the higher levels which then adjust the scoring weights of the lower levels to meet the higher level targets.

15 Further work

With reference to Fig. 15.1 which shows how the adaptive elements could integrate into the scheduler, there are a number of things which need implementing before an Adaptive Look-Ahead Scheduler (ALAS) can be used for operational scheduling. Several of the specialized modules must be designed and written and the search mechanism must be speeded up.

Although I have looked at how stability and other environmental conditions affect the scheduling process, I have assumed we can actually determine this information. In fact such prediction is not easy.

External forecasting sources We already have access to and use internal weather and sky monitoring systems. Ideally however, we should be able to obtain information from various external sources:- short and medium range weather forecasts are available from the European Centre for Medium Range Weather Forecasting (ECMWF). Atmospheric seeing and extinction forecasts are available from MeteoBlue. Marchant (2009) has performed an analysis on the accuracy of MeteoBlue predictions of weather parameters for several days ahead and found around 70% agreement over that timescale.

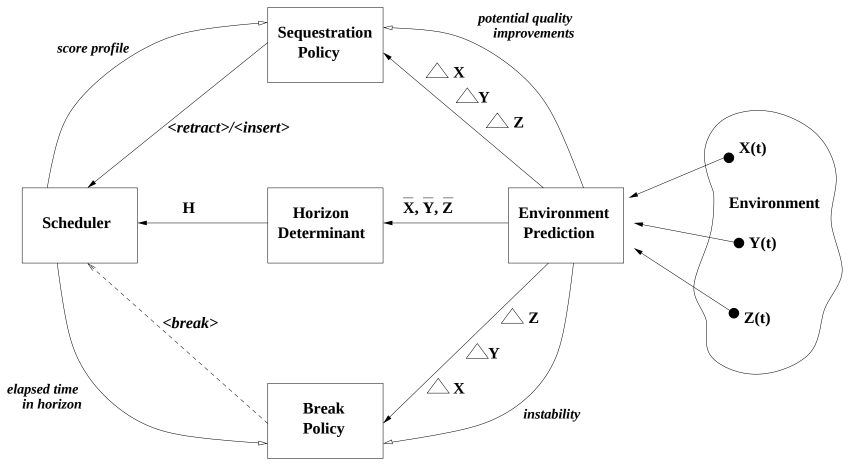


Figure 15.1: Simplified architecture to allow environmental prediction to influence the scheduler. The *HorizonDeterminant* uses details of stability to decide on the current horizon. Volatile events (typically a surge of new observations) are fed to the *SequestrationPolicy* to decide whether to modify the running schedule by retracting existing commitments and inserting new ones. Changes to environmental stability are fed to the *BreakPolicy* to determine whether to stop the current schedule and reschedule or to carry on with reduced value.

Sensor fusion There is potential for a significant amount of work on fusing these sources information together to make stability predictions. The forecast information exists on different timescales which might point to different sources being applicable to the different planning timescales. We should not ignore the option to use human input. Humans are good at extracting predictive information from very complex datasets - it could prove most effective to have an interactive input to the scheduler where an interpreter sets out the probabilities of bad weather based on detailed study of a variety of forecast sources.

Texture based metrics In terms of measuring and more importantly predicting complexity there may be some advantage in looking at texture measurements as described by Beck et al. (1997).

Both QLAS and ELAS are very inefficient due to the random solution generation/search process. This is not particularly important in a simulation context - other than in the time taken to perform the simulations. However, in an operational context where solutions need to be found quickly to avoid holding up the execution, more efficient and intelligent solution generation and search mechanisms would be required.

Solution generator and search mechanism There are several heuristic techniques which might be used to speed up the generation of solutions to test.

- *Breeder* Genetic techniques might be used to breed solutions. In any potential sequence solution there will be good (high scoring) segments and poorer (low scoring) segments. In some cases the good part might be early in the sequence while in another it might be nearer the end. If we were to take the good parts of one solution and merge these with the good parts of another solution we could create a new solution with more good parts then either parent. Similarly, we might find that by changing just a few components in an already good solution we could achieve a better solution. These two techniques represent in effect the genetic mechanisms of *crossover* and *mutation*. There are potential options to speed this mechanism up by farming out the breeding to a number of seperate *breeder stations* running in parallel on different hosts.
- *Energy minimization* This technique assumes that each group excerts a repulsive force on all surrounding groups which might be in competition for the same region of the time axis. The force excerted would be related to the importance and urgency of the group (its score). *Strong* groups would be able to claim their ideal location on the timeline whereas *weak* groups would be forced off into less desirable regions (from their point of view) or even ejected if a strong group required the whole of the weak group's feasible window. Ultimately the system should settle down into a *minimum energy* configuration.

So far I have assumed that the selection of the *best* sequence should be calculated in terms of the sequence for which the highest potential reward is calculated. Other factors such as *robustness* to changing conditions and *flexibility* might be included in this decision.

Break policy Notifications of instability need to be integrated into the system so that decisions can be made as to when to curtail the execution of the current sequence and recalculate.

Sequestration policy To take advantage of improvements in the conditions or the arrival of better quality observations, a sequestration policy must be devised to allow the executing schedule to be re-calculated or modified.

Retraction heuristics When responding to a sequestration, the retraction heuristic determines the choice of group to replace.

Expected Future Reward I have already briefly discussed the possibility of using Expected Future Reward (EFR) in the context of determining whether to schedule a group now or at some later point based on the likelihood of suitable conditions. This technique allows one to *hedge one's bets* against future uncertainty by discounting the return from decisions made in the present. A study by Sozou (1998) reveals that humans typically use a form of discounting where the perceived hazard rate changes with time yielding a hyperbolic discount rate. They provide mechanisms whereby a discount rate can be calculated based on non-occurance of a perceived risk using Bayesian updating and also note that such non-exponential time-preference curves can cross-over so that preferences may change. The main research interest in this area would be in the determination of a suitable discount rate with reference to predicted and observed stability and volatility measures.

There are a number of areas in which the Phase 2 model could be enhanced to provide additional features.

- User preferences. So far, the scoring mechanism has been biased toward enterprise measures of reward (Sect. 3.3) and little regard has been taken of the individual user's measures of value. It is true that for instance the relative elevation of targets is taken into account to ensure that targets are observed at *good* times, however we do not allow the trade-offs between the various user-preferences to be made. The Phase 2 model could be enhanced to allow it to represent user's own quantification of these tradeoffs.

- Quality requirements. There might be some merit in allowing users to specify limits on the amount, regularity and quality of data taken. Currently when a group fails for whatever reason it becomes reschedulable and no charge is made. In some cases, most of the required observations might have been made anyway. In such situations it would increase effciency if a QOS measure could be used to determine the degree of success rather than the present hard cutoff rule.

Part III

Appendices

A Phase 2 model generator and its configuration

The Phase 2 model generator can be used to create ODB snapshots according to various distributions and with varying degrees of complexity. The Phase 2 model is generated with the following parameters:-

A.1 Notation

The following distributions are used throughout:-

- $U[a, b]$ a uniform random number selected from the range $[a, b]$.
- $G(\mu, \sigma)$ a random number with probability distributed as a gaussian with mean μ and standard deviation σ.

A.2 Configuration parameters

Root Specifies the root name for this phase2 model. This is the name by which the model is accessed via a Phase2ModelProvider.

Name The name of the site.

Φ Latitude of the site.

L Longitude of the site.

T specifies the *current* point in the semester. Typically this time coincides with the start time of a simulation run.

ΔT_B represents the time difference from T to the start of the current semester.

ΔT_F represents the time difference from T to the end of the current semester.

N_P controls the total number of proposals generated. These will have activation times distributed according to $U[\Delta T_B, T]$ and expiry dated distributed according to $U[T, \Delta T_F]$.

N_G controls the number of groups per proposal, distributed as $U[1, N_G]$.

N_O controls the number of observations per group, distributed as $U[1, N_O]$.

Groups are generated using various timing constraint classes according to the following fractions:-

g_{flex} Fraction of groups to be generated with Flexible timing constraints.

g_{mon} Fraction of groups to be generated with monitor timing constraints.

g_{int} Fraction of groups to be generated with minimum interval timing constraints.

All groups are constructed with start and expiry dates selected from the range $U[\Delta T_B, T]$ and $U[T, \Delta T_F]$ respectively.

For monitor groups, the periods are selected with probability determined by variable f_{m_i} from one of a set of 3 normal distributions $G(\mu_{m_i}, \sigma_{m_i})$ defined by variables:-

μ_{m_i} mean value for monitor period distribution i.

σ_{m_i} standard-deviation for monitor period distribution i

The window fraction is taken from the distribution $U[0.25, 1.0]$.

For MinimumInterval groups, the minimum window is selected with probability f_{i_i} from one of 3 normal distributions $G(\mu_{i_i}, \sigma_{i_i})$ defined by variables:-

μ_{i_i} mean value for minimum interval length distribution i.

σ_{i_i} standard-deviation for minimum interval lengthdistribution i

Observation exposure times are selected with probability e_i from one of 2 exposure length distributions $E(\mu_{e_i}, \sigma_{e_i})$ defined by variables:-

μ_{e_i} mean value for exposure length distribution i.

σ_{e_i} standard-deviation for exposure length distribution i.

N_n exposure-max-count specifies the maximum number of multruns per observation, the number of multruns is taken from $U[1, N_m]$.

The budget available to each proposal is taken from $U[B_{min}, B_{max}]$ with used fraction distributed as $U[U_{min}, U_{max}]$.

Proposals have a scientific priority rating taken from the range 0 (HIGH) to 3 (LOW). The allocation is specified accroding to the fractions:-

P_0 Fraction of proposals with scientific priority 0 (HIGH).

P_1 Fraction of proposals with scientific priority 0 (MED).

P_2 Fraction of proposals with scientific priority 0 (LOW).

Groups have internal priorities selected from the following set.

G_1 Fraction of groups with priority level 1 (normal)

G_2 Fraction of groups with priority level 2 (raised)

G_3 Fraction of groups with priority level 3 (medium)

G_4 Fraction of groups with priority level 4 (high)

G_5 Fraction of groups with priority level 5 (urgent)

G_{BGR} Fraction of groups with Background priority level

G_{STD} Fraction of groups which represent photometric standards (nominal priority 3).

Various observing constraints are represented by the following parameters:-

f_{Dark} Fraction of groups which require Dark lunar conditions.

f_{Photom} Fraction of groups which require photometric extinction conditions.

f_{Poor} Fraction of groups which require minimum seeing of *poor* ($seeing > 1.3''$).

f_{Av} Fraction of groups which require minimum seeing of *average* ($0.8'' < seeing < 1.3''$).

f_{Ex} Fraction of groups which require minimum seeing of *excellent* ($seeing < 0.8''$).

B Supporting statistics

This appendix contains tables of statistical data derived from the datasets throughout the experimental part of the thesis. References to any associated figure(s) are shown in the table captions.

Mean (μ) and standard deviation (σ) values.				
Model (N_g)	μ_C	σ_C	μ_{SU}	σ_{SU}
S (100)	2.69	1.02	46.26	7.39
L (300)	8.8	2.94	54.37	5.64
M (900)	14.0	2.35	60.86	2.21
H (2500)	18.8	3.51	65.97	3.7

Table B.1: Statistics of the variation of Q_{SU} with C_{DC} based on data from Fig. 9.3. A linear least squares model $Q_{SU} = a + bC_C$ was derived after 5 iterations giving $a = 44.2 \pm 2$% and $b = 1.124 \pm 6$% with $\chi^2 = 23.32$.

Mean (μ) and standard deviation (σ) values.				
Model (N_g)	μ_C	σ_C	μ_{XT}	σ_{XT}
S (100)	3.39	1.65	0.81	0.18
L (300)	9.0	2.8	0.85	0.05
M (900)	14.99	2.21	0.91	0.08
H (2500)	17.8	4.1	0.95	0.038

Table B.2: Statistics of the variation of Q_{XT} with C_{DC} based on data from Fig. 9.4.

Fraction of nights on which schedulers beat base scheduler.			
Selection model	ζ_{best}	ζ_{RS}	ζ_{FR}
ζ_{best}	-	62%	59.5%
ζ_{RS}	38%	-	44%
ζ_{FR}	40.5%	56%	-

Table B.3: Fraction of nights on which scheduler with given selection model beats other models based on data displayed in Figs. 10.2, 10.3, 10.4. The entry in column c and row r denotes the fraction of nights that scheduler c beats scheduler r.

Amount by which scheduler beats base model.			
Selection model	ζ_{best}	ζ_{RS}	ζ_{FR}
ζ_{best}	-	8.53	7.28
ζ_{RS}	5.76	-	7.66
ζ_{FR}	6.8	8.75	-

Table B.4: Amount $\Delta_{Q_{SU}}$ by which scheduler with given selection model beats other models based on data displayed in Figs.10.2, 10.3, 10.4. The entry in column c and row r denotes the amount by which scheduler c beats scheduler r.

The following tables show statistics of improvement in specified quality metrics between baseline scheduler implementations and one or more comparative schedulers. The main statistic derived is Δ_{90}, the relative fractional improvement and its 90% confidence limits.

Values of Q_{el} versus w_{el}			
w_{el}	0.1	0.5	1.0
BDS_{random}	0.57 (±0.04)	0.57 (±0.04)	0.57 (±0.04)
BDS_{best}	0.57 (±0.02)	0.57 (±0.18)	0.89 (±0.02)
Δ_{90}(BDS_{best},BDS_{random})	0.18% (±10.01%)	0.35% (±41.70%)	57.24% (±10.43%)

Table B.5: Comparison of schedulers (BDS_{best}) relative to baseline (BDS_{random}) based on results displayed in Fig. 7.3 . Measurements show values of Q_{el} and its relative improvement against selected values of w_{el}

Values of Q_{el} versus w_{el}			
w_{el}	0.1	0.5	1.0
BDS	0.57 (±0.02)	0.57 (±0.18)	0.89 (±0.02)
H1	0.73 (±0.00)	0.73 (±0.00)	0.73 (±0.00)
Δ_{90}(H1,BDS)	29.45% (±4.29%)	29.23% (±40.56%)	-17.53% (±3.31%)

Table B.6: Comparison of schedulers (H1) relative to baseline (BDS) based on results displayed in Fig. 7.3 . Measurements show values of Q_{el} and its relative improvement against selected values of w_{el}

Values of Q_{px} versus w_{el}			
w_{el}	0.0	0.5	0.9
BDS_{random}	1.29 ($\pm$0.13)	1.29 ($\pm$0.13)	1.29 ($\pm$0.13)
BDS_{best}	1.70 ($\pm$0.02)	1.78 ($\pm$0.02)	1.67 ($\pm$0.02)
$\Delta_{90}(BDS_{best}, BDS_{random})$	31.78% ($\pm$13.04%)	37.91% ($\pm$13.07%)	29.38% ($\pm$13.00%)

Table B.7: Comparison of schedulers (BDS_{best}) relative to baseline (BDS_{random}) based on results displayed in Fig. 8.2a . Measurements show values of Q_{px} and its relative improvement against selected values of w_{el}

Values of Q_{px} versus w_{el}			
w_{el}	0.0	0.5	0.9
BDS_{random}	1.20 ($\pm$0.12)	1.20 ($\pm$0.12)	1.20 ($\pm$0.12)
BDS_{best}	1.76 ($\pm$0.03)	1.81 ($\pm$0.01)	1.69 ($\pm$0.01)
Δ_{90}(BDS_{best},BDS_{random})	47.28% ($\pm$12.70%)	51.46% ($\pm$12.39%)	41.42% ($\pm$12.40%)

Table B.8: Comparison of schedulers (BDS_{best}) relative to baseline (BDS_{random}) based on results displayed in Fig. 8.2b. Measurements show values of Q_{px} and its relative improvement against selected values of w_{el}

Values of Q_{oa} versus w_{el}			
w_{el}	0.1	0.55	0.95
BDS_{random}	0.83 (±0.02)	0.83 (±0.02)	0.83 (±0.02)
BDS_{best}	0.86 (±0.00)	0.85 (±0.00)	0.92 (±0.00)
$\Delta_{90}(BDS_{best}, BDS_{random})$	3.06% (±2.98%)	2.34% (±2.98%)	10.61% (±2.95%)

Table B.9: Comparison of schedulers (BDS_{best}) relative to baseline (BDS_{random}) based on results displayed in Fig. 8.3a . Measurements show values of Q_{oa} and its relative improvement against selected values of w_{el}

Values of Q_{oa} versus w_{el}			
w_{el}	0.1	0.55	0.95
BDS_{random}	0.82 ($\pm$0.02)	0.82 ($\pm$0.02)	0.82 ($\pm$0.02)
BDS_{best}	0.84 ($\pm$0.00)	0.84 ($\pm$0.01)	0.94 ($\pm$0.00)
$\Delta_{90}(BDS_{best}, BDS_{random})$	3.18% ($\pm$2.90%)	3.18% ($\pm$2.95%)	14.67% ($\pm$2.89%)

Table B.10: Comparison of schedulers (BDS_{best}) relative to baseline (BDS_{random}) based on results displayed in Fig. 8.3b . Measurements show values of Q_{oa} and its relative improvement against selected values of w_{el}

Values of Q_{oa} versus w_{el}				
w_{el}	0.0	0.1	0.6	0.95
BDS_{random}	0.79 (±0.02)	0.79 (±0.02)	0.79 (±0.02)	0.79 (±0.02)
BDS_{best}	0.76 (±0.01)	0.93 (±0.02)	0.92 (±0.02)	0.92 (±0.00)
$\Delta_{90}(BDS_{best}, BDS_{random})$	-3.18% (±3.98%)	18.22% (±4.50%)	17.71% (±4.39%)	17.07% (±3.33%)

Table B.11: Comparison of schedulers (BDS_{best}) relative to baseline (BDS_{random}) based on results displayed in Fig. 8.4a . Measurements show values of Q_{oa} and its relative improvement against selected values of w_{el}

Values of Q_{oa} versus w_{el}				
w_{el}	0.0	0.1	0.6	0.95
BDS_{random}	0.81 ($\pm$0.02)	0.81 ($\pm$0.02)	0.81 ($\pm$0.02)	0.81 ($\pm$0.02)
BDS_{best}	0.76 ($\pm$0.02)	0.90 ($\pm$0.01)	0.92 ($\pm$0.02)	0.96 ($\pm$0.01)
Δ_{90}(BDS_{best},BDS_{random})	-6.05% ($\pm$4.25%)	11.60% ($\pm$3.53%)	13.83% ($\pm$4.47%)	18.02% ($\pm$3.40%)

Table B.12: Comparison of schedulers (BDS_{best}) relative to baseline (BDS_{random}) based on results displayed in Fig. 8.4b . Measurements show values of Q_{oa} and its relative improvement against selected values of w_{el}

Values of Q_{px} versus w_{el}			
w_{el}	0.1	0.65	0.9
BDS_{random}	0.17 ($\pm$0.02)	0.17 ($\pm$0.02)	0.17 ($\pm$0.02)
BDS_{best}	0.09 ($\pm$0.02)	0.11 ($\pm$0.02)	0.18 ($\pm$0.02)
$\Delta_{90}(BDS_{best},BDS_{random})$	-46.11% ($\pm$19.04%)	-35.93% ($\pm$17.89%)	7.78% ($\pm$16.81%)

Table B.13: Comparison of schedulers (BDS_{best}) relative to baseline (BDS_{random}) based on results displayed in Fig. 8.5a . Measurements show values of Q_{px} and its relative improvement against selected values of w_{el}

Values of Q_{px} versus w_{el}			
w_{el}	0.1	0.65	0.9
BDS_{random}	0.41 (±0.02)	0.41 (±0.02)	0.41 (±0.02)
BDS_{best}	0.12 (±0.03)	0.14 (±0.01)	0.23 (±0.02)
$\Delta_{90}(BDS_{best},BDS_{random})$	-71.91% (±9.55%)	-66.83% (±5.99%)	-44.55% (±7.26%)

Table B.14: Comparison of schedulers (BDS_{best}) relative to baseline (BDS_{random}) based on results displayed in Fig. 8.5b. Measurements show values of Q_{px} and its relative improvement against selected values of w_{el}

Values of Q_{td} versus w_{el}			
w_{el}	0.0	0.55	0.95
BDS_{random}	3.30 (±0.65)	3.30 (±0.65)	3.30 (±0.65)
BDS_{best}	4.48 (±0.41)	5.07 (±0.53)	3.59 (±0.35)
Δ_{90}(BDS_{best},BDS_{random})	35.76% (±29.81%)	53.64% (±32.53%)	8.79% (±28.69%)

Table B.15: Comparison of schedulers (BDS_{best}) relative to baseline (BDS_{random}) based on results displayed in Fig. 8.6a . Measurements show values of Q_{td} and its relative improvement against selected values of w_{el}

Values of Q_{td} versus w_{el}			
w_{el}	0.0	0.55	0.95
BDS_{random}	3.29 (±0.70)	3.29 (±0.70)	3.29 (±0.70)
BDS_{best}	5.37 (±0.47)	4.91 (±0.37)	3.99 (±0.45)
$\Delta_{90}(BDS_{best},BDS_{random})$	63.22% (±32.72%)	49.24% (±30.80%)	21.28% (±32.38%)

Table B.16: Comparison of schedulers (BDS_{best}) relative to baseline (BDS_{random}) based on results displayed in Fig. 8.6b. Measurements show values of Q_{td} and its relative improvement against selected values of w_{el}

Values of Q_{rn} versus w_{el}			
w_{el}	0.1	0.6	0.95
BDS_{random}	29.60 (±3.60)	29.60 (±3.60)	29.60 (±3.60)
BDS_{best}	47.04 (±1.38)	47.67 (±1.24)	39.45 (±1.28)
$\Delta_{90}(BDS_{best},BDS_{random})$	58.92% (±16.67%)	61.05% (±16.47%)	33.28% (±16.52%)

Table B.17: Comparison of schedulers (BDS_{best}) relative to baseline (BDS_{random}) based on results displayed in Fig. 8.7a. Measurements show values of Q_{rn} and its relative improvement against selected values of w_{el}

Values of Q_{rn} versus w_{el}			
w_{el}	0.1	0.6	0.95
BDS_{random}	27.80 ($\pm$3.50)	27.80 ($\pm$3.50)	27.80 ($\pm$3.50)
BDS_{best}	50.85 ($\pm$1.07)	49.93 ($\pm$1.21)	39.49 ($\pm$1.13)
Δ_{90}(BDS_{best},BDS_{random})	82.91% ($\pm$16.85%)	79.60% ($\pm$17.05%)	42.05% ($\pm$16.93%)

Table B.18: Comparison of schedulers (BDS_{best}) relative to baseline (BDS_{random}) based on results displayed in Fig. 8.7b . Measurements show values of Q_{rn} and its relative improvement against selected values of w_{el}

Values of Q_{SU} versus C_c			
C_c	1.0	15.0	25.0
ζ_{Best}	52.64 ($\pm$3.74)	65.00 ($\pm$3.74)	69.17 ($\pm$3.74)
ζ_{RS}	46.62 ($\pm$2.47)	66.51 ($\pm$2.47)	71.77 ($\pm$2.47)
$\Delta_{90}(\zeta_{RS},\zeta_{Best})$	-11.44% ($\pm$10.90%)	2.32% ($\pm$8.83%)	3.77% ($\pm$8.29%)
Q_1	47.92 ($\pm$3.96)	69.43 ($\pm$3.96)	73.55 ($\pm$3.96)
$\Delta_{90}(Q_1,\zeta_{Best})$	-8.98% ($\pm$13.24%)	6.82% ($\pm$10.72%)	6.33% ($\pm$10.08%)
Q_2	53.30 ($\pm$2.93)	72.31 ($\pm$2.93)	76.80 ($\pm$2.93)
$\Delta_{90}(Q_2,\zeta_{Best})$	1.24% ($\pm$11.55%)	11.25% ($\pm$9.36%)	11.04% ($\pm$8.79%)
Q_4	53.10 ($\pm$2.39)	76.99 ($\pm$2.39)	82.37 ($\pm$2.39)
$\Delta_{90}(Q_4,\zeta_{Best})$	0.88% ($\pm$10.79%)	18.46% ($\pm$8.74%)	19.09% ($\pm$8.21%)

Table B.19: Comparison of schedulers (ζ_{RS},Q_1,Q_2,Q_4) relative to baseline (ζ_{Best}) based on results displayed in Figs. 9.5, 9.6, 9.7, 9.8, 9.9, 9.10. Measurements show values of Q_{SU} and its relative improvement against selected values of C_c

Values of Q_{SU} versus τ_E			
τ_E	15.0	180.0	480.0
ζ_{Best}	72.50 ($\pm$7.50)	77.50 ($\pm$11.00)	77.50 ($\pm$13.50)
ζ_{FR}	76.20 ($\pm$9.00)	75.20 ($\pm$9.00)	76.20 ($\pm$9.00)
Δ_{90}(ζ_{FR},ζ_{Best})	5.10% ($\pm$20.68%)	-2.97% ($\pm$23.47%)	-1.68% ($\pm$26.80%)
ζ_{RS}	81.70 ($\pm$6.50)	73.40 ($\pm$4.50)	74.70 ($\pm$5.50)
Δ_{90}(ζ_{RS},ζ_{Best})	12.69% ($\pm$17.52%)	-5.29% ($\pm$19.63%)	-3.61% ($\pm$24.08%)

Table B.20: Comparison of schedulers (ζ_{FR},ζ_{RS}) relative to baseline (ζ_{Best}) based on results displayed in Figs. 10.7, 10.8, 10.9 . Measurements show values of Q_{SU} and its relative improvement against selected values of τ_E

Values of Q_{SU} versus n_e			
n_e	0.0	6.0	12.0
BDS	118.00 (±4.00)	118.00 (±4.00)	118.00 (±4.00)
$Q_{0.5}$	134.10 (±0.10)	108.31 (±3.75)	104.40 (±4.80)
Δ_{90}($Q_{0.5}$,BDS)	13.64% (±4.34%)	-8.21% (±5.95%)	-11.53% (±6.78%)
Q_2	128.20 (±0.10)	109.20 (±4.20)	108.10 (±3.20)
Δ_{90}(Q_2,BDS)	8.64% (±4.34%)	-7.46% (±6.29%)	-8.39% (±5.56%)
Q_4	136.20 (±0.10)	111.50 (±4.10)	110.70 (±1.76)
Δ_{90}(Q_4,BDS)	15.42% (±4.34%)	-5.51% (±6.21%)	-6.19% (±4.74%)

Table B.21: Comparison of schedulers ($Q_{0.5}$,Q_2,Q_4) relative to baseline (BDS) based on results displayed in Fig. 11.1 . Measurements show values of Q_{SU} and its relative improvement against selected values of n_e

Values of Q_{SU} versus proximity					
proximity	0.0	30.0	90.0	210.0	360.0
$QLAS_{0.5}$	1.00 (±0.07)	0.91 (±0.08)	0.83 (±0.09)	0.62 (±0.09)	0.42 (±0.08)
$ELAS_{0.5}$	1.02 (±0.07)	0.92 (±0.09)	0.86 (±0.09)	0.67 (±0.08)	0.37 (±0.08)
$\Delta_{90}(ELAS_{0.5},QLAS_{0.5})$	1.70% (±12.40%)	1.32% (±16.38%)	3.51% (±19.62%)	8.08% (±24.87%)	-11.81% (±33.81%)

Table B.22: Comparison of schedulers ($ELAS_{0.5}$) relative to baseline ($QLAS_{0.5}$) based on results displayed in Fig. 12.7 . Measurements show values of Q_{SU} and its relative improvement against selected values of proximity

Values of Q_{SU} versus proximity					
proximity	0.0	30.0	90.0	210.0	360.0
$QLAS_1$	0.74 (±0.09)	0.98 (±0.08)	0.82 (±0.09)	0.61 (±0.10)	0.36 (±0.08)
$ELAS_1$	0.85 (±0.08)	0.96 (±0.08)	0.84 (±0.08)	0.59 (±0.09)	0.38 (±0.08)
$\Delta_{90}(ELAS_1,QLAS_1)$	15.22% (±21.16%)	-2.74% (±15.36%)	2.93% (±17.86%)	-3.30% (±28.84%)	5.52% (±40.52%)

Table B.23: Comparison of schedulers ($ELAS_1$) relative to baseline ($QLAS_1$) based on results displayed in Fig. 12.8 . Measurements show values of Q_{SU} and its relative improvement against selected values of proximity

Values of Q_{SU} versus proximity					
proximity	0.0	30.0	90.0	210.0	360.0
$QLAS_2$	0.38 (±0.08)	0.54 (±0.09)	0.81 (±0.09)	0.68 (±0.08)	0.44 (±0.09)
$ELAS_2$	0.53 (±0.09)	0.69 (±0.10)	0.79 (±0.09)	0.63 (±0.09)	0.44 (±0.09)
$\Delta_{90}(ELAS_2,QLAS_2)$	39.89% (±38.59%)	27.57% (±30.86%)	-2.84% (±20.00%)	-7.11% (±23.10%)	0.46% (±36.12%)

Table B.24: Comparison of schedulers ($ELAS_2$) relative to baseline ($QLAS_2$) based on results displayed in Fig. 12.9 . Measurements show values of Q_{SU} and its relative improvement against selected values of proximity

Values of Q_{SU} versus proximity					
proximity	0.0	30.0	90.0	210.0	360.0
$QLAS_4$	0.17 (±0.06)	0.20 (±0.07)	0.41 (±0.09)	0.70 (±0.09)	0.35 (±0.09)
$ELAS_4$	0.39 (±0.09)	0.39 (±0.08)	0.57 (±0.10)	0.60 (±0.08)	0.39 (±0.08)
$\Delta_{90}(ELAS_4,QLAS_4)$	135.33% (±86.73%)	96.45% (±69.89%)	37.05% (±40.61%)	-14.06% (±21.44%)	10.86% (±43.56%)

Table B.25: Comparison of schedulers ($ELAS_4$) relative to baseline ($QLAS_4$) based on results displayed in Fig. 12.10 . Measurements show values of Q_{SU} and its relative improvement against selected values of proximity

Bibliography

A. Allan, T. Naylor, I. A. Steele, T. Jenness, B. Cavanagh, F. Economou, E. Saunders, A. Adamson, J. Etherton, and C. Mottram. eSTAR: intelligent observing and rapid responses. In H. Lewis & G. Raffi, editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 5496, pages 313–322, September 2004.

A. Allan, T. Naylor, and E. S. Saunders. The eSTAR network - agent architectures for astronomy. *Astronomische Nachrichten*, 327:767–+, September 2006.

B. Apolloni, C. Carvalho, and D. de Falco. Quantum stochastic optimization. *Stochastic Processes and their Applications*, 33(2):233 – 244, 1989.

ARI. Liverpool telescope instrument pipelines. webpage, 2011. URL `http://telescope.livjm.ac.uk/Info/TelInst/Pipelines/`.

J. Christopher Beck, Andrew J. Davenport, Edward M. Sitarski, and Mark S. Fox. Texture-based heuristics for scheduling revisited. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97)*, pages 241–248, Providence, Rhode Island, USA, 1997. AAAI Press / MIT Press. ISBN 0-262-51095-2.

John E. Bell and Patrick R. McMullen. Ant colony optimization techniques for the vehicle routing problem. *Advanced Engineering Informatics*, 18(1):41–48, 2004.

Chris Benn and Sara Ellison. La palma night-sky brightness. Technical report, Isaac Newton Group, may 1998. ING La Palma Technical Note 115.

James R. Bitner and Edward M. Reingold. Backtrack programming techniques. *Commun. ACM*, 18(11):651–656, 1975.

J. Bresina, M. Drummond, and K. Swanson. Managing action duration uncertainty with just-in-case scheduling, 1994.

J. Bresina, K. Golden, D. Smith, and R. Washington. Increased flexibility and robustness of mars rovers. In *Proceedings of the Fifth International. Symposium on Artificial Intelligence, Robotics and Automation in Space*, Noordwijk, Netherlands, 1999.

J.L. Bresina. Heuristic-biased stochastic sampling. In *Proceedings of the AAAI-96*, pages 271–278, 1996.

S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau. Integrated planning and execution for autonomous spacecraft. In *Aerospace Conference, 1999. Proceedings. 1999 IEEE*, volume 1, pages 263–271, 1999a.

S. Chien, G. Rabideau, R. Knight, R. Sherwood, B. Engelhardt, T. Estlin D. Mutz, B. Smith, F. Fisher, T. Barrett, G. Stebbins, and D. Tran. ASPEN automating space mission operations using automated planning and scheduling, 2000.

Steve Chien, Russell Knight, Andre Stechert, Rob Sherwood, and Gregg Rabideau. Using iterative repair to increase the responsiveness of planning and scheduling for autonomous spacecraft. In *IJCAI99 Workshop on Scheduling and Planning to meet Real-time Monitoring in a Dynamic and Uncertain World*, Stockholm, Sweden, August 1999b.

Hon Wai Chun and Rebecca Y. M. Wong. N*–an agent-based negotiation algorithm for dynamic scheduling and rescheduling. *Advanced Engineering Informatics*, 17 (1):1 – 22, 2003.

Vincent Cicirello and Stephen Smith. Randomizing dispatch scheduling policies. In *The 2001 AAAI Fall Symposium: Using Uncertainty Within Computation*, November 2001.

Vincent Cicirello and Stephen Smith. Amplification of search performance through randomization of heuristics. In *Principles and Practice of Constraint Programming: 8th International Conference, Proceedings*, volume LNCS 2470 of Lecture Notes in Computer Science, pages 124–138. Springer-Verlag, September 2002.

S. Darmoul, H. Pierreval, and S.H. Gabouj. Scheduling using artificial immune system metaphors: A review. In *Service Systems and Service Management, 2006 International Conference on*, volume 2, pages 1150 –1155, 2006.

A.J. Davenport, C. Gefflot, and J.C. Beck. Slack-based techniques for robust schedules. In *Proceedings of the Sixth European Conference on Planning (ECP-2001)*, 2001.

M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, 1992.

M. Dorigo and L.M. Gambardella. Ant colony system: A cooperative learning approach to the travelling salesman problem. Technical report, IRIDIA, Universite Libre de Bruxelles, 1995.

Jrgen Dorn, Roger Kerr, and Gabi Thalhammer. Reactive scheduling: improving the robustness of schedules and restricting the effects of shop floor disturbances by fuzzy reasoning. *International Journal of Human-Computer Studies*, 42(6):687 – 704, 1995.

Mark Drummond, John Bresina, and Keith Swanson. Just-in-case scheduling. In *AAAI'94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 2)*, pages 1098–1104, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence. ISBN 0-262-61102-3.

M.S. Fox, N. Sadeh, and C. Baykan. Constrained heuristic search. In *Proc.IJCAI*, pages 309–316. International Joint Conference on Artificial Intelligence, August 1989.

A. S. Fraser. Simulation of genetic systems by automatic digital computers. i. introduction. *Aust. J. Biol. Sci*, 10:484–491, 1957.

S. N. Fraser and I. A. Steele. Observation scheduling simulation framework: design and first results. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7019 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, August 2008.

Stephen N. Fraser. Robotic telescope scheduling: The Liverpool Telescope experience. In Peter J. Quinn and Alan Bridger, editors, *Optimizing Scientific Return for Astronomy through Information Technologies, Proceedings of SPIE*, volume 5493, pages 331–340. SPIE, sep 2004.

Stephen N. Fraser and Iain A. Steele. Object oriented design of the Liverpool Telescope Robotic Control System. In Hilton Lewis, editor, *Advanced Telescope and Instrumentation Control Software II, Proceedings of SPIE*, volume 4848, pages 443–454. SPIE, dec 2002.

D. L Fried. Optical resolution through a randomly inhomogeneous medium for very long and very short exposures. *Journal of the Optical Society of America*, 56(10): 1372–1379, Oct 1966.

Alex F. Fukunaga, Gregg Rabideau, and Steve Chien. Robust local search for spacecraft operations using adaptive noise. In *IWPSS 4th International Workshop on Planning and Scheduling for Space*, June 2004.

Luca Maria Gambardella, ric Taillard, and Giovanni Agazzi. Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In *New Ideas in Optimization*, pages 63–76. McGraw-Hill, 1999.

F. Glover. Tabu search and adaptive memory programming - advances, applications, and challenges. In R.V. Helgason R.S. Barr and J.L. Kenngington, editors, *Interfaces in Computer Science and Operations Research*, pages 1–75. Kluwer Academic Publishers, Hingham, MA, USA, 1996.

F. Glover, M. Laguna, and R. Mart. *Advances in Evolutionary Computation: Theory and Applications*, chapter Scatter Search, pages 519–537. Springer-Verlag, 2003.

P.P. Grass. La reconstruction du nid et les coordina- tions interindividuelles chez bellicositermes natalensis et cubitermes sp. la thorie de la stigmergie. *Insectes Sociaux*, pages 41–81, 1959.

M.A. Guerrero, R.J. Garcia-Lopez, R.L.M. Corradi, A. Jimenez, J.J. Fuensalida, J.M. Rodriguez-Espinosa, A. Alonso, M. Centurion, and F. Prada. Extinction over the canarian observatories: the limited influence of saharan dust. *New Astronomy Reviews*, 42:529–532, 1998.

E. Hart, P. Ross, and J. Nelson. Producing robust schedules via an artificial immune system. In *ICEC*, pages 464–469. IEEE, 1998.

E. Hart, P. Ross, and J. Nelson. Scheduling chicken catching – an investigation into the success of a genetic algorithm on a real world scheduling problem. *Annals Of Operations Research*, 92(0):363–380, Jan 1999.

Emma Hart and Peter Ross. An immune system approach to scheduling in changing environments. In *GECCO*, pages 1559–1566, 1999.

P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4 (2):100 –107, 1968.

Arnoldo C. Hax and Harlan C. Meal. Hierarchical integration of production planning and scheduling. Technical report, 1973.

H. Hoos. An adaptive noise mechanism for walksat, 2002.

Toru Ishida and Masashi Shimbo. Improving the learning efficiencies of realtime search. In *AAAI/IAAI, Vol. 1*, volume 1, pages 305–310, 1996.

Y. Ishida. Fully distributed diagnosis by pdp learning algorithm: towards immune network pdp model. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 777 –782 vol.1, June 1990.

Niels K. Jerne. The generative grammar of the immune system. *Bioscience Reports*, 5:439–451, 1985.

N.K. Jerne. Towards a network theory of the immune system. *Annales d'immunologie*, 125c:373–389, Jan 1974.

Mark D Johnston and Glen E Miller. Spike: Intelligent scheduling of hubble space telescope observations. In Mark Fox and Monty Zweben, editors, *Intelligent Scheduling*, pages 391–422. Morgan-Kaufmann, San Francisco, CA, USA, 1994.

A. Jones and J. Rabelo. Survey of job shop scheduling techniques, 1998.

J. Kennedy. The particle swarm: social adaptation of knowledge. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 303 –308, apr 1997.

James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proc. IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

D L King. Atmospheric extinction at the roque de los muchachos observatory, la palma. Technical report, Isaac Newton Group, sept 1985.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

Laurence Kramer and Stephen Smith. Maximizing flexibility: A retraction heuristic for oversubscribed scheduling problems. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, August 2003. prepublication copy - Copyright International Joint Conference On Artificial Intelligence, Inc.

Laurence Kramer and Stephen Smith. Task swapping for schedule improvement: A broader analysis. In *Proceedings 14th International Conference on Automated Planning and Scheduling*, June 2004.

K. Krisciunas and B. E. Schaefer. A model of the brightness of moonlight. *Publications of the Astronomical Society of the Pacific*, 103:1033–1039, 1991.

Osepayshvili Anna Reeves Daniel M. MacKie-Mason, Jeffrey K. and Michael P Wellman. Price prediction strategies for market-based scheduling. In *18th International Conference on Automated Planning and Scheduling*, June 2004.

J.M. Marchant. Weather forecast analysis. Published on internal ARI web, 2009.

Jonathan Marchant, Robert J. Smith, and Iain A. Steele. Calibration of the boltwood cloud sensor. In *Proceedings of the Society of Photooptical Instrumentation Engineers*, volume 7012, page 70123U. SPIE, June 2008.

David Mcallester, Bart Selman, and Henry Kautz. Evidence for invariants in local search. In *In Proceedings of AAAI-97*, pages 321–326, 1997.

Martijn Mes, Matthieu van der Heijden, and Aart van Harten. Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems. *European Journal of Operational Research*, 181(1):59 – 75, 2007.

Steve Minton, Mark D Johnston, Andrew B Philips, and Philip Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

Kazua Miyashita and Masahiro Hori. A Distributed Architecture for Planning and Scheduling that Learns through Negotiation Cases. In *IEEE International Conference on Emerging Technologies and Factory Automation*, 1996.

Kazuo Miyashita and Katia P. Sycara. CABINS: A Framework of Knowledge Acquisition and Iterative Revision for Schedule Improvement and Reactive Repair. *Artificial Intelligence*, 76(1-2):377–426, 1995.

K. Mori, M. Tsukiyama, and T. Fukuda. Immune algorithm and its application to factory load dispatching planning. In *Japan-USA Symposium on flexible automation*, pages 1343–1346, 1994.

C. Mottram. High level interfaces to the Robonet-1.0 Homogeneous Telescope Network. *Astronomische Nachrichten*, 327:806–+, September 2006.

C Muñoz-Tuñón, J Vernin, and A.M. Varela. Night-time image quality at roque de los muchachos observatory. *Astronomy and Astrophysics Supplement Series*, pages 183–193, October 1997.

C. Muñoz-Tuñón, A. M. Varela, and T. Mahoney. Homogeneity of image quality at the Roque de los Muchachos Observatory. *New Astronomy Review*, 42:409–416, November 1998.

Seshahayee Murthy, Rama Akkiraju, John Rachlin, and Frederick Wu. Agent-based cooperative scheduling. In *In Proceedings of AAAI Workshop on Constraints and Agents*, pages 112–117. AAAI Press, 1997.

Nicola Muscettola. Scheduling by iterative partition of bottleneck conflicts. Technical Report CMU-RI-TR-92-05, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, February 1992.

T. Naylor, A. Allan, and I. A. Steele. Heterogenous telescope networks: An introduction. *Astronomische Nachrichten*, 327:741–+, September 2006.

H. Van Dyke. Parunak. Go to the ant: Engineering principles from natural multi-agent systems. *Annals of Operations Research*, (75):69–101, 1997.

Christopher G. Pernin, Katherine Comanor, Lance Menthe, Louis R. Moore, and Tim Andersen. Allocation of forces, fires, and effects using genetic algorithms. Technical report, RAND Corporation, 2008.

N. Policella, S. F. Smith, A. Cesta, and A. Oddi. Steps toward computing flexible schedules. In *Proceedings of Online-2003 Workshop CP 2003*, pages 39–53, 2003.

Nicola Policella. *Scheduling with uncertainty a proactive approach using partial order schedules*. PhD thesis, Universita degli studi di Roma (La Sapienza), Via Salaria 113, I-00198, Roma, Italy, 2005.

J Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

G. Rabideau, R. Knight, S. Chien, A. Fukunaga, and A. Govindjee. Iterative repair planning for spacecraft operations in the aspen system, 1999.

Gregg Rabideau, Barbara Engelhardt, and Steve A. Chien. Using generic preferences to incrementally improve plan quality. In *AIPS*, pages 236–245, 2000.

Simone C. Riedmiller and Martin A. Riedmiller. A neural reinforcement learning approach to learn local dispatching policies in production scheduling. In *IJCAI*, pages 764–771, 1999.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003. ISBN 0137903952.

Norman Sadeh. Look ahead techniques for micro-opportunistic job shop scheduling. Technical Report CMU-CS-91-102, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, March 1991.

Norman Sadeh, Katia Sycara, and Yalin Xiong. Backtracking techniques for the job shop scheduling constraint satisfaction problem. Technical Report CMU-RI-TR-94-31, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, October 1994.

E. Sanmart, A. Huercio, A. Espua, and L. Puigjaner. A combined scheduling/reactive scheduling strategy to minimize the effect of process operations uncertainty in batch plants. *Computers and Chemical Engineering*, 20(2):1263 – 1268, 1996.

M. Sarazin and F. Roddier. The ESO differential image motion monitor. *Astronomy and Astrphysics*, 227:294–300, 1990.

E. S. Saunders, T. Naylor, and A. Allan. Optimal placement of a limited number of observations for period searches. *Astronomy and Astrophysics*, 455:757–763, August 2006.

E. S. Saunders, T. Naylor, and A. Allan. An autonomous adaptive scheduling agent for period searching. *Astronomische Nachrichten*, 329:321–+, March 2008.

Michael J Shaw, Sang Chan Park, and Narayan Raman. Intelligent scheduling with machine learning capabilities: The induction of scheduling knowledge. Technical Report CMU-RI-TR-90-25, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, nov 1990.

R. J. Smith, N. R. Clay, S. N. Fraser, J. M. Marchant, C. M. Moss, and I. A. Steele. Switching the Liverpool Telescope from a full-service operating model to self-service. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7737 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, 2010.

Stephen Smith. Reactive scheduling systems. In D.E. Brown and W.T. Scherer, editors, *Intelligent Scheduling Systems*. Kluwer Press, 1995.

Stephen Smith and C. Cheng. Slack-based heuristics for constraint satisfaction scheduling. In *Proceedings 11th National Conference on Artificial Intelligence*, July 1993.

Stephen Smith, O. Lassila, and Marcel Becker. Configurable, mixed-initiative systems for planning and scheduling. In A. Tate, editor, *Advanced Planning Technology*. AAAI Press, May 1996. ISBN 0-929280-98-9.

Stephen Smith, Marcel Becker, and Laurence Kramer. Continuous management of airlift and tanker resources: A constraint-based approach. *Mathematical and Computer Modeling – Special Issue on Defense Transportation: Algorithms, Models and Applications for the 21st Centry*, 39(6-8):581–598, 2004.

Stephen F. Smith and Ora Lassila. Toward the development of flexible mixed-initiative scheduling tools. In *ARPA-Rome Laboratory Planning Initiative Workshop*, 1994.

P. Sorensen. Ing meteorological and seeing archive. web, 2002. URL `http://catserver.ing.iac.es/weather/archive/index.php`.

P.D. Sozou. On hyperbolic discounting and uncertain hazard rates. In *Proceedings of the Royal Society B Biological Sciences*, volume 265, pages 2015–2020, 1998.

I. A. Steele. The Liverpool telescope. *Astronomische Nachrichten*, 325:519–521, October 2004.

Iain A Steele and David Carter. Control software and scheduling of the liverpool robotic telescope. In Hilton Lewis, editor, *Proceedings of SPIE*, volume 3112 of *Telescope Control Systems II*, pages 222–233, sep 1997.

Katia Sycara, Dajun Zheng, and Kazuo Miyashita. Using case-based reasoning to acquire user scheduling preferences that change over time. In *Proc. of the Eleventh IEEE Conference on Artificial Intelligence for Application*. IEEE, 1995.

E. D. Taillard, L. Gambardella, M. Gendreau, and J. Potvin. Adaptive memory programming: A unified view of meta-heuristics. Technical report, Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale, 1998.

J Timmis, T Knight, L N De Castro, and E Hart. An overview of artificial immune systems. In R Paton, H Bolouri, M Holcombe, J H Parish, and R Tateson, editors, *"Computation in Cells and Tissues: Perspectives and Tools for Thought"*, Natural Computation Series, pages 51–86. Springer, November 2004. ISBN 3540003584.

Y. Tsapras, R. Street, K. Horne, C. Snodgrass, M. Dominik, A. Allan, I. Steele, D. M. Bramich, E. S. Saunders, N. Rattenbury, C. Mottram, S. Fraser, N. Clay, M. Burgdorf, M. Bode, T. A. Lister, E. Hawkins, J. P. Beaulieu, P. Fouqué, M. Albrow, J. Menzies, A. Cassan, and D. Dominis-Prester. RoboNet-II: Follow-up observations of microlensing events with a robotic network of telescopes. *Astronomische Nachrichten*, 330, January 2009.

A. M. Varela, C. Bertolin, C. Muñoz-Tuñón, J. J. Fuensalida, and S. Ortolani. In situ calibration using satellite data results. In *Revista Mexicana de Astronomia y Astrofisica Conference Series*, volume 31 of *Revista Mexicana de Astronomia y Astrofisica, vol. 27*, pages 106–112, October 2007.

J. Vernin and C. Muñoz-Tuñón. The temporal behaviour of seeing. *New Astronomy Review*, 42:451–454, November 1998.

J. Vernin and C. Munoz-Tunon. Optical seeing at la palma observatory. *Astronomy and Astrophysics*, 257:811–816, 1992.

Makuto Yokoo and Toru Ishida. Sesrch algorithms for agents. In Gerhard Weiss, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 166–199. The MIT Press, Cambridge, MA, USA, 1999.

W. Zhang and T. G. Dietterich. A reinforcement learning approach to job-shop scheduling. In *Proceedings of the International Joint Conference on Artificial Intellience*, 1995.

Monty Zweben, Brian Daun, Eugene Davis, and Michael Deale. Scheduling and rescheduling with iterative repair. In Mark Fox and Monty Zweben, editors, *Intelligent Scheduling*, pages 241–255. Morgan-Kaufmann, San Francisco, CA, USA, 1994. ISBN 1-55860-260-7.