N95-23742

# Robust Telescope Scheduling

**Keith Swanson** NASA

**John Bresina** Recom Technologies

**Mark Drummond** Recom Technologies

AI Research Branch, Mail Stop: 269-2
NASA Ames Research Center, Moffett Field, CA 94035-1000
*E-mail*: {swanson, bresina, drummond}@ptolemy.arc.nasa.gov
*Phone*: 415-604-6016, *Fax*: 415-604-3594

## KEY WORDS AND PHRASES

Automatic telescopes, execution errors, Just-In-Case scheduling, robust execution, telescope scheduling.

## ABSTRACT

This paper presents a technique for building robust telescope schedules that tend not to break. The technique is called Just-In-Case scheduling and it implements the common sense idea of being prepared for likely errors, just in case they should occur. The JIC algorithm analyzes a given schedule, determines where it is likely to break, reinvokes a scheduler to generate a contingent schedule for each highly probable break case, and produces a "multiply contingent" schedule. The technique was developed for an automatic telescope scheduling problem, and the paper presents empirical results showing that Just-In-Case scheduling performs extremely well for this problem.

## INTRODUCTION

This paper presents and evaluates a technique for generating schedules that have robust execution behavior. The technique is called Just-In-Case scheduling, or JIC, and it implements the common sense idea of being prepared for likely execution errors, just in case they should occur. JIC handles schedule execution errors that are due to the presence of actions with uncertain durations. JIC was developed as part of a larger telescope management and scheduling project. This section outlines only key aspects of the problem; more details are available elsewhere [4; 6].

In this application domain, the telescopes are land-based and fully automatic; a telescope control computer opens the observatory at twilight and collects data through the night without human assistance (see Genet and Hayes [8] for details). We are implementing an overall automated management system [6] to enable participating astronomers to submit observation requests and obtain results from a remotely located telescope. This interaction occurs *via* electronic communication networks, without the necessity of human intervention. To ensure the telescope load is balanced over weeks or months, the system will also include a sophisticated long-term loader [2]. The long-term loader will assign observation requests to specific nights. Each night, the assigned observations are given to a scheduler to determine the specific times during the night they are to be executed. Producing *robust* nightly schedules is the role of JIC.

An observation request specifies both hard constraints and soft preferences. The most important hard constraint is the *observing window*. Each observation request, or action, can begin execution only in a specific interval of time within a night; this interval is defined by the astronomer who submitted the request. In the remainder of this paper we refer to each observation request as an *action*.

The scheduling problem is to synthesize a schedule that satisfies all hard constraints and achieves a good score according to an objective function based on the soft preferences. A schedule is a sequence of actions, each with an *enablement interval* assigned by the scheduler. The assigned enablement interval of each action is a subinterval of the action's (astronomer-provided) observing window. A scheduler assigns the enablement intervals to further restrict when the actions will begin execution. This paper does not address the problem of generating a basic schedule (this is discussed in [7]) – we assume the existence of a scheduler that, given a set of requests and an objective function, produces a feasible observing schedule with a reasonable score. (Our current scheduler produces reasonable schedules in less than one minute.)

In this domain, a typical action has a duration of several minutes with action duration uncertainty occurring due to mechanical slop in the telescope drive train, software timing inaccuracies, and star centering. The amount of time it takes to center a star depends on how accurately the telescope is pointed when it starts the search and how clear the sky is while the centering is going on. Hence, it is impossible to accurately predict the duration of an observing action. All we can do is gather data from actual executions and then calculate the mean and standard deviation of each action's duration. Observation actions are typically executed many times over a period of weeks or months, so such statistics are easily available.

The telescope used in this application is fully automatic and runs unattended; thus, unlike many scheduling domains where printing a schedule is the final goal, our system must be able to automatically execute a schedule. Schedule execution proceeds by executing each action in the scheduled sequence. After an action finishes execution, if the current time is outside of the next action's (scheduler-assigned) enablement interval, then the schedule breaks and execution halts.

Schedule breakage is the central problem. The predicted start time of an action in a schedule is based on the sum of the estimated durations of the actions that precede it. Hence, the further into the future an action occurs in the schedule, the greater the uncertainty surrounding its actual start time. Given the way that uncertainty grows, it is possible for a schedule to break due solely to accumulated duration prediction errors.

There is a simple solution to the problem of duration prediction errors: make the start time of each action equal to a worst case estimate of the previous action's finish time and introduce a busy-wait in case the previous action finishes early. Unfortunately, introducing such busy-waits will waste observing time. Our goal is to avoid schedule breaks without sacrificing schedule quality.

Schedules can also fail for reasons other than duration prediction uncertainty. Clouds or wind can make star acquisition impossible, resulting in unavoidable schedule breaks. In our system, when the schedule breaks, the telescope invokes the scheduler to generate a new one for the current situation. Hence, while weather can cause a break in schedule execution, the system is robust enough to dynamically reschedule and try again. Dynamic rescheduling could also be used, in place of JIC, to handle breaks due to duration prediction errors. However, the problem with dynamic rescheduling is that it wastes valuable observing time whenever the telescope controller is waiting for a schedule. There is limited observing time available during the night, and we do not want to waste it!

JIC proactively manages duration uncertainty by identifying high probability schedule breaks and, for each one, generating an alternative schedule just in case the break occurs during execution. This proactive management can use off-line time during the day to compute and store alternative schedules in order to reduce on-line rescheduling time during the night. JIC produces a "multiply contingent" schedule that specifies what actions the telescope controller should take, conditioned by the current situation. Thus, if accumulated duration prediction errors force the telescope into a situation for which the nominal (*i.e.*, the initial) schedule is inapplicable, then an appropriate contingent schedule (if available) is automatically selected and execution continues. If an appropriate schedule is not available, the system resorts to dynamic rescheduling.

## JUST-IN-CASE SCHEDULING

In overview, the JIC algorithm accepts a schedule as input and robustifies it as follows. First, using a model of how action durations can vary, the temporal uncertainty at each step in the schedule is estimated. Second, the most probable break due to this uncertainty is determined. Third, the break point is "split" into two hypothetical cases: one in which the schedule breaks and one in which it does not. Fourth, the scheduler is invoked on a new scheduling subproblem to produce an alternative schedule for the break case. Fifth, this alternative schedule is integrated with the initial schedule producing an updated multiply contingent schedule. This completes consideration of one break case; if there is more time before schedule execution begins, then the JIC process can be repeated with the current multiply contingent schedule as the new input.

Each action $A_i$ has a duration mean $\mu_i$ and standard deviation $\sigma_i$. One of the preconditions of each action is the interval of time during which it can begin execution; let $P_i$ be this *precondition interval* for $A_i$. (The precondition interval for observation requests is provided by an astronomer.)

A *schedule* is a sequence of actions, where each action is associated with an *enablement interval*, $E_i$, assigned by the scheduler: $(A_0, E_0); \ldots; (A_n, E_n)$, such that for $i = 0, \ldots, n$, $E_i \subseteq P_i$. During schedule execution, as soon as action $A_{i-1}$ is finished executing, action $A_i$ is selected for enablement testing; $A_i$ is enabled if the current time is within $E_i$. If $A_i$ is enabled, then it is immediately executed; else, the schedule breaks.

A *multiply contingent schedule* can be thought of as a set of alternative schedules; to save space, our implementation uses a tree to represent this set of schedules. Let $\beta(i)$ be defined such that $A_{\beta(i)}$ is the predecessor of $A_i$ in the schedule, if one exists. For simplicity, we assume that $A_0$ is the unique first action in a multiply contingent schedule.

Using a duration uncertainty model discussed below, JIC estimates the temporal uncertainty at each step in the schedule by starting at the beginning of the schedule and propagating uncertainty forward. This process involves estimating the time at which each action in the schedule will start and finish executing. The *start interval*, $S_i$, is the set of possible execution start times for action $A_i$. Similarly, the *finish interval*, $F_i$, is the set of possible execution finish times for action $A_i$. Let $S_0$ denote the interval during which schedule execution can start. For our telescope application, schedule execution always starts exactly at twilight; hence, $S_0$ is the degenerate interval [twilight, twilight].

$A_i$ cannot start executing at a time outside its enablement window. Hence, if $A_{\beta(i)}$ finishes executing at a time outside of $E_i$, then either an action in an alternative contingent schedule will be executed or the schedule will break. Thus, $S_i$ is computed to be $F_{\beta(i)} \cap E_i$.

Given that $A_i$'s start interval, $Si$, is $[t_1, t_2]$, its finish interval, $F_i$, is computed to be $[t_1 + \mu_i - \sigma_i, t_2 + \mu_i + \sigma_i]$. The current duration uncertainty model simply uses one standard deviation of the mean when computing each finish interval – this has worked well in practice, as shown by the empirical results in the next section.

The break probability of an action is a function of the enablement probability of that action and of all preceding actions. Let $p(\text{enable}(A_i))$ be the ***enablement probability*** for action $A_i$; that is, the probability that $A_i$ will be enabled when selected. It is computed to be the proportion of the previous action's finish interval during which $A_i$ is enabled.

$$p(\text{enable}(A_i)) = \frac{|F_{\beta(i)} \cap E_i|}{|F_{\beta(i)}|}$$

For simplicity, this computation is based on the erroneous assumption that all of an action's possible finish times are equi-probable (*i.e.*, that $F_{\beta(i)}$ has a uniform probability distribution) and, hence, is only an estimate of the true enablement probability.

Let $p(\text{select}(A_i))$ be the ***selection probability*** for action $A_i$; that is, the probability that $A_i$ will be selected for enablement testing. An action will be selected if the preceding action was both selected and enabled; the schedule's first action will always be selected.

For $i = 0$: $p(\text{select}(A_i)) = 1.0$.
For $i > 0$: $p(\text{select}(A_i)) = p(\text{select}(A_{\beta(i)})) \times p(\text{enable}(A_{\beta(i)}))$.

Let $p(\text{break}(A_i))$ be the ***break probability*** for action $A_i$; that is, the probability that the schedule will break at $A_i$ when it is selected for enablement testing.

$$p(\text{break}(A_i)) = p(\text{select}(A_i)) \times [1 - p(\text{enable}(A_i))].$$

Note that the computation of break probabilities is similar to the computation of the conditional probabilities characterized by a Markov chain.

After determining the action with the highest break probability, JIC "splits" the associated uncertainty time interval into two subintervals. One subinterval will be the intersection of the finish interval $F_{\beta(i)}$ with the enablement interval $E_i$. The remaining subinterval (not part of the intersection) is split off as a break case and a new scheduling subproblem is formed using a point in the subinterval as the start time. JIC then invokes the scheduler on this subproblem and incorporates the returned alternative schedule into the original schedule.

## EMPIRICAL EVALUATION

To evaluate the performance of JIC we performed an experiment using real telescope scheduling data. (Additional experimental results and algorithm details are available elsewhere [3; 5].) The observation actions were provided by Greg Henry of Tennessee State University [9; 11]. The scheduler used in this experiment deterministically hill-climbs on a domain-specific heuristic [1]. The experiment required collecting data from thousands of schedule executions; since this is impractical on a real telescope, we developed a simulator of the telescope controller's schedule executor. The simulator computes an action's execution duration by using a random variable with a normal (Gaussian) probability distribution whose mean and standard deviation are exactly those characterized by statistics obtained from a number of nights of actual execution on a telescope at the Fairborn Observatory (Mt. Hopkins, Arizona).

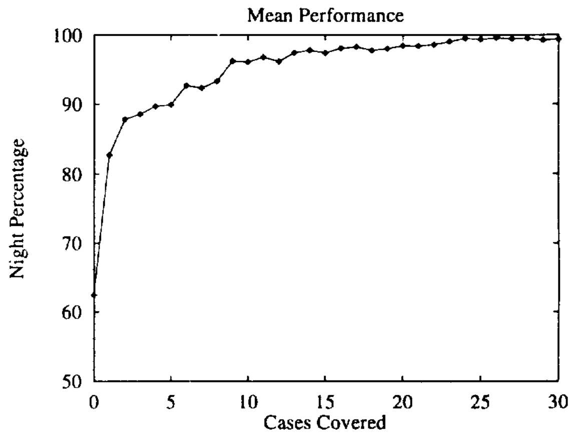

Figure 1: Mean performance, measured as night percentage, *vs.* cases covered.

The experimental question is: given real telescope scheduling data, can JIC provide a useful increase in schedule robustness within a reasonable number of contingent cases? To answer this question we varied the number of break cases considered and measured how far into the night a multiply contingent schedule would execute without dynamic rescheduling. The experimental procedure is as follows.

First, the scheduler is used to find a single nominal schedule. This schedule is executed 1000 times in the simulator; for each execution run we note the percentage of the night that the schedule executes before halting, either due to a break or schedule completion. Next, we allow JIC to find and fix what it deems to be the next most probable break case. We then run the augmented schedule through the execution simulator (again, 1000 times). In this manner, we allow JIC to cover up to thirty break cases.

Figure 1 illustrates the resulting performance. The independent variable is the number of break cases covered by JIC. The dependent variable is the percentage of the night that the schedule executes before halting, averaged over 1000 runs. It clearly shows that the mean percentage of the night executed increases with the number of cases considered by JIC. The performance increase is most dramatic early on, as we had hoped. After only ten cases, the schedule executes, on average, through 96% of the night. Although not shown, experimental results also indicate that schedule size (measured as the total number of actions contained in the multiply contingent schedule) increases linearly with the number of cases, as one might expect.

## CONCLUSION

This paper has presented an algorithm for Just-In-Case scheduling. Using almost any scheduler and simple statistical models of duration uncertainty, the algorithm proactively makes a nominal schedule more robust. Despite some rather egregious modeling assumptions, the algorithm works extremely well for a real telescope scheduling problem. Traditional intuitions surrounding the management of uncertain action outcomes suggest the inevitability of large search spaces and intractable reasoning. Using a "splitting" technique, our algorithm makes action outcome distinctions only when necessary (see Hanks [10] for background to this idea). Further, most of the likely schedule breaks are covered in a few iterations of JIC.

While JIC works extremely well for our particular telescope scheduling problem, it will not necessarily fare so well on all domains. We have analyzed the nature of the schedule breaks in our domain in order to characterize the general conditions under which JIC achieves useful robustness increments in a few iterations. The results are suggestive, but not yet mathematically precise. Essentially, JIC appears to work well when the following three conditions hold.

First, there must be room for improvement. If the prior probability of successful execution of the schedule is close to 1.0, there is not much JIC can add. Second, there must be a small number of schedule breaks responsible for most of the total break probability mass. If this is so, then each break case covered by JIC stands a good chance of increasing the probability of executing the entire schedule. Third, each contingent schedule found must be no worse in its break characteristics than the nominal schedule. In some sense, this is simply a recursive application of the first two conditions; it requires that each contingent schedule be as easy to robustify as the initial one.

Finally, we recognize that a number of interesting issues remain outstanding regarding the applicability of JIC to other domains and how JIC compares to, or might integrate with, other existing scheduling techniques. This is an excellent area for further work.

## ACKNOWLEDGEMENTS

Significant thanks to Will Edgington for helping make the telescope management and scheduling software a reality.

## REFERENCES

[1] Boyd, L., Epand, D., Bresina, J., Drummond, M., Swanson, K., Crawford, D., Genet, D., Genet, R., Henry, G., McCook, G., Neely, W., Schmidtke, P., Smith, D., and Trublood, M. 1993. Automatic Telescope Instruction Set 1993. *International Amateur Professional Photoelectric Photometry Communications*, No. 52.

[2] Bresina, J. 1994. Telescope Loading: A Problem Reduction Approach. In *Proc. of i-SAIRAS 94*. JPL, Pasadena, CA.

[3] Bresina, J., Drummond, M., Swanson, K. 1994. Managing Action Duration Uncertainty with Just-In-Case Scheduling. *Proceedings of the AAAI Spring Symposium on Decision-Theoretic Planning*. Stanford, CA. March, 1994.

[4] Bresina, J., Drummond, M., Swanson, K., and Edgington, W. 1994. Automated Management and Scheduling of Remote Automatic Telescopes. *Astronomy for the Earth and Moon*, the proceedings of the 103rd Annual Meeting of the Astronomical Society of the Pacific, D. Pyper Smith (ed.).

[5] Drummond, M., Bresina, J., and Swanson, K. 1994. Just-In-Case Scheduling. In *Proc. of AAAI-94*.

[6] Drummond, M., Bresina, J., Swanson, K., and Edgington, W. 1994. The Associate Principal Astronomer Telescope Operations Model. In *Proc. of i-SAIRAS 94*. JPL, Pasadena, CA.

[7] Drummond, M., Swanson, K., and Bresina, J. (*in press*). Robust Scheduling and Execution for Automatic Telescopes. In *Intelligent Scheduling*, M. Fox & M. Zweben (eds). Morgan-Kaufmann.

[8] Genet, R.M., and Hayes, D.S. 1989. *Robotic Observatories: A Handbook of Remote-Access Personal-Computer Astronomy*. AutoScope Corporation, Mesa, AZ.

[9] Hall, D. S. and Henry, G. W. 1992. Performance Evaluation of Two Automatic Telescopes after Eight Years. *Automated Telescopes for Photometry and Imaging*, S. J. Adelman, R. J. Dukes, and C. J. Adelman (eds.) (San Francisco: Astronomical Society of the Pacific), p. 13.

[10] Hanks, S. 1990. Practical Temporal Projection. In *Proc. of AAAI-90*. pp. 158 – 163.

[11] Henry, G. W. and Hall, D. S. (*in press*) The Quest for Precision Robotic Photometry. *International Amateur Professional Photoelectric Photometry (IAPPP) Communications* 55.