Just-In-Case Scheduling

Mark Drummond Recom Technologies
John Bresina Recom Technologies
Keith Swanson NASA
AI Research Branch, Mail Stop: 269-2
NASA Ames Research Center
Moffett Field, CA 94035-1000 USA
e-mail: {drummond, bresina, swanson}@ptolemy.arc.nasa.gov

Abstract

This paper presents an algorithm, called *Just-In-Case Scheduling*, for building robust schedules that tend not to break. The algorithm implements the common sense idea of being prepared for likely errors, just in case they should occur. The Just-In-Case algorithm analyzes a given nominal schedule, determines the most likely break, and reinvokes a scheduler to generate a contingent schedule to cover that break. After a number of iterations, the Just-In-Case algorithm produces a "multiply contingent" schedule that is more robust than the original nominal schedule. The algorithm has been developed for a real telescope scheduling domain in order to proactively manage schedule breaks that are due to an inherent uncertainty in observation durations. The paper presents empirical results showing that the algorithm performs extremely well on a representative problem from this domain.

Introduction

This paper presents and evaluates an algorithm for generating schedules that have robust execution behavior. The algorithm is called *Just-In-Case Scheduling*, or JIC, and it implements the common sense idea of being prepared for likely errors, just in case they should occur. JIC handles schedule execution errors that are due to the presence of actions with uncertain durations.

In modeling terms, an action with uncertain duration is *stochastic*: its outcome cannot be uniquely determined. It is commonly believed that if stochastic actions are included in a planning or scheduling formalism, then the resulting reasoning problem will be intractable. If the average stochastic branching factor is b, then at each branch point, a nominal schedule covers one of the outcomes and $b-1$ outcomes remain as possible execution breaks, or errors. Thus, a nominal schedule containing n actions has approximately $n \times (b-1)$ different possible errors. Proactively managing each of these errors would mean finding a schedule to cover each possible error case. Further, the new schedule for each error case would itself produce new possible errors that also require proactive management. Under these conditions, the number of possible error cases grows rapidly, and the problem is clearly intractable.

The results presented in this paper run counter to these expectations. We show that for an extremely large and practical problem, contingent reasoning about stochastic actions is not only tractable, but is efficient and effective. The JIC algorithm performs extremely well for our telescope scheduling application domain, and we have reason to believe that it should work well on similar domains.

In the remainder of the paper, we first introduce the application domain, then define the JIC algorithm, then present an empirical evaluation of that algorithm, and lastly conclude with some general remarks.

The Domain

Just-In-Case scheduling has been developed for a real telescope scheduling domain. This section outlines only key aspects of the domain; more details are available elsewhere: Bresina, *et al.* (1993), Bresina *et al.* (*in press*), Genet (1994), and Genet & Hayes (1989).

In this domain, telescope users electronically submit observation requests to a central location for subsequent scheduling. The requests contain "hard" constraints, defined by basic physics, and a number of "soft" preferences. The most important hard constraint is an *observing window*. Each observation request can be executed only in a specific time window. A window is an interval of time, typically between one and eight hours, defined by the astronomer who submitted the request. Once submitted, an observation request can be active for weeks or months. In the remainder of the paper, we refer to each observation request as an *action*.

The scheduling problem is one of finding a sequence of actions that satisfies all hard constraints completely and that achieves a good score according to an objective function that measures how well the schedule satisfies 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 can begin execution. This paper does not address the problem of finding a schedule (discussed by Drummond, Swanson, & Bresina, *in press*) – we assume the existence of a scheduler that produces a feasible and reasonable-scoring observing schedule, given a set of actions, constraints, and an objective function.

Finding a schedule is only the first step. The telescope used in this domain is fully automatic and runs unattended; thus, unlike many scheduling domains where printing a schedule is the final goal, the system must be able to automatically execute a schedule. A schedule is executed 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.

Execution of a typical action involves repetitions of the following three-step pattern: first, move the telescope to point at (or near) a star; second, search a limited section of the sky in order to center the star within the telescope lens; third, take an instrument reading. The amount of time it takes to center a star depends on how accurately the telescope is pointed when it starts the centering search and how clear the sky is. The star centering search process makes it impossible to predict exactly how long each action will take to execute.

Schedule breakage due to uncertain action duration is the central problem addressed by JIC. 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 into the future, it is possible for a schedule to call for an action to be executed at a time outside its scheduler-assigned interval. Hence, a schedule can break during execution solely because of 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 wastes valuable observing time during the night. In the long run, the amount of time wasted during busy-waits can be significant. Our goal is to avoid schedule breaks without wasting valuable observing time.

Schedules fail for reasons other than duration uncertainty. Clouds or wind can make star centering impossible, resulting in unavoidable schedule breakage. In our system, when the current schedule breaks, the telescope controller invokes the scheduler to generate a new schedule. Thus, while weather can cause a break in schedule execution, the system is robust enough to dynamically reschedule and try again. The problem with on-line rescheduling is that it wastes valuable observing time whenever the telescope is idle, waiting for the scheduler. There is limited observing time available during the night, and we do not want to waste it. Further motivation for the development of JIC is given in Swanson, Bresina, & Drummond (1994).

The basic idea behind JIC is to proactively manage execution breaks caused by action duration uncertainty. Proactive error management uses off-line time during the day to compute and store alternative schedules in order to reduce on-line rescheduling time during the night.

JIC defines a "wrapper" algorithm that allows one to repeatedly and proactively call an existing scheduler. This approach is extremely general in the sense that, by decoupling scheduling from reasoning about uncertainty, it allows an arbitrary scheduler to employ JIC. The scheduler used to date in our system does not exploit any information regarding action duration uncertainty. This scheduler is simpler and possibly more efficient than one that does take action duration uncertainty into account. However, schedulers which ignore uncertainty might find schedules that are intrinsically difficult to make robust. An alternative approach is to have the scheduler consider both the objective function score obtained by a given schedule *and* the action duration uncertainty. Given that action duration uncertainty can be modeled probabilistically, decision theory tells us how to combine the objective function and probability measures in terms of expected utility. Unfortunately, decision theory does not provide a computationally effective means of doing so for problems with large search spaces. Previous experiments have indicated that a typical search space for our telescope domain contains on the order of 10^{57} different schedules (Drummond, Swanson, & Bresina, *in press*). Thus, using a separate mechanism to manage duration uncertainty seems to be a reasonable compromise.

The Algorithm

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 possible 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. We now consider each step in more detail.

In order to model the execution duration for each action, we keep statistics from actual executions at the telescope. Each day, we derive an updated duration mean and standard deviation for each action. We believe that an action's execution duration has a normal (gaussian) distribution. However, for reasons of simplicity and efficiency, we model duration uncertainty as a uniform distribution in the current implementation. Our experimental results show that this approximation works quite well in practice.

In order to explain the detailed steps of the algorithm, we first need to define some terms. Each action A_i has a duration mean μ_i and standard deviation σ_i. One of the preconditions of each action is the interval of time during which it can begin execution; let W_i be this *observing window* for A_i. (Recall that the observing window 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 $i = 0, \ldots, n$, $E_i \subseteq W_i$. 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; otherwise, 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.

Using the duration uncertainty model, 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 simplicity, let us assume that schedule execution always starts exactly at twilight; hence, S_0 is the degenerate interval [twilight, twilight].

A_i cannot start executing 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 implementation simply uses one standard deviation of the mean when computing each finish interval, and this has worked well in practice.

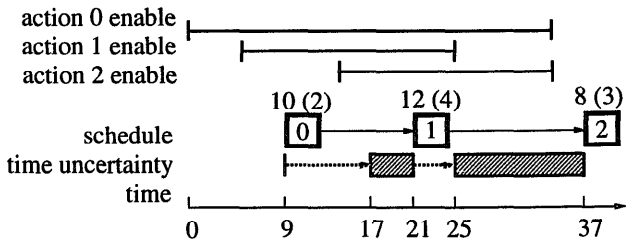

Figure 1: Propagation of temporal uncertainty.

See Figure 1 for a stylized schedule containing three actions labeled 0, 1, and 2. The enablement intervals assigned by the scheduler are given at the top of the figure, and the schedule ordering is indicated by solid arrows between the actions. The numbers above an action indicate a mean and standard deviation for that action's duration (*e.g.*, action 0 has a mean duration of 10 and a standard deviation of 2). The schedule is predicted to start exactly at time 9, indicated by a degenerate uncertainty interval above time 9 (the solid line above time 9). JIC computes a finish interval for action 0 that ranges from 17 to 21 and a finish interval for action 1 that ranges from 25 to 37.

Using the finish interval estimates and the assigned enablement intervals of the actions, JIC determines the action in the schedule that has the highest probability of breaking. 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.

$$\text{For } i = 0: \quad p(\text{enable}(A_i)) = 1.0$$
$$\text{For } i > 0: \quad p(\text{enable}(A_i)) = \frac{|F_{\beta(i)} \cap E_i|}{|F_{\beta(i)}|}$$

For simplicity, this computation is based on the erroneous assumption that all of an action's possible finish times are equally likely (*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.

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

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.

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

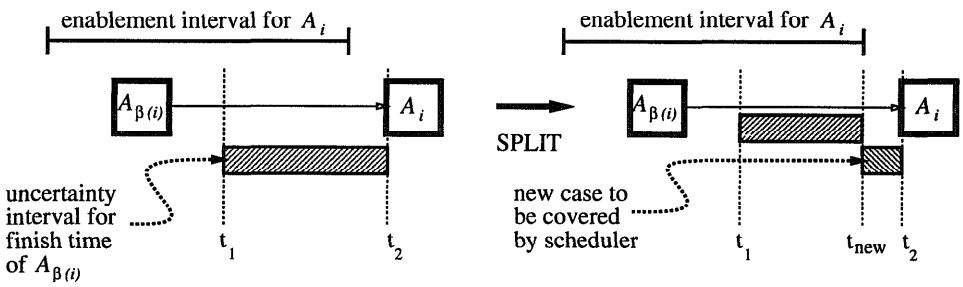

Figure 2: Splitting an uncertainty interval.

Note that the computation of break probabilities is similar to the computation of conditional probabilities in a Markov chain (Thiebaux, *et al.*, 1993).

After determining the action with the highest break probability, JIC splits the associated uncertainty time interval into two subintervals. This is shown graphically in Figure 2. A_i is the action identified as having the highest break probability, and $A_{\beta(i)}$ is the previous action in the schedule. The possible finish interval of $A_{\beta(i)}$, $[t_1, t_2]$, is split according to how it overlaps A_i's assigned enablement interval. The subinterval $[t_{new}, t_2]$ is split off as a break case, since it is outside the enablement interval. A new scheduling subproblem is formed with t_{new} as its start time. JIC then invokes the scheduler on this subproblem and incorporates the returned alternative schedule into the original schedule. The resulting multiply contingent schedule still contains $A_{\beta(i)}$ followed by A_i. The alternative schedule will be executed only if after execution of $A_{\beta(i)}$, A_i is not enabled. In our tree representation of a multiply contingent schedule, each branch point corresponds to an uncertainty interval that has been split.

Suppose that the telescope management system stops accepting new observation actions one hour before twilight. This gives the scheduler one hour to find a schedule for that night. It is relatively easy to find a high-scoring schedule in about one minute (Drummond, Swanson, & Bresina, *in press*). This leaves roughly 59 minutes for JIC to proactively make the schedule more robust. JIC incurs overhead to find the most probable schedule break and to create a new scheduling subproblem. However, if we assume that the overhead time required for JIC is comparatively small and that each call JIC makes to the scheduler takes about one minute, then there is time available to consider about fifty possible break cases.

The time cost of the JIC algorithm is proportional to the size of the multiply contingent schedule, and schedule size grows linearly with the number of cases covered by JIC. Since the algorithm's time cost is a function of the number of cases covered, it is natural to wonder how many cases must be covered to usefully increase the execution robustness of a typical observing schedule. This is precisely the question addressed by the first experiment presented in the next section.

Empirical Evaluation

To evaluate the performance of JIC we performed several experiments using real telescope scheduling data. The observation actions were provided by Greg Henry of Tennessee State University (Hall & Henry, 1992; Henry & Hall, *in press*). The scheduler used in these experiments deterministically hill-climbs on a domain-specific heuristic (Boyd, *et al.*, 1993). The experiments 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. In the first experiment, 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 set equal to the statistics obtained from a number of nights of actual execution on a telescope at the Fairborn Observatory (Mt. Hopkins, Arizona).

The 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 measured how far into the night a multiply contingent schedule executes before rescheduling would be required. The experimental procedure is as follows.

First, the scheduler is used to find a single nominal schedule. This schedule is executed 1000 times by 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 most probable break case, and then run the augmented schedule through the execution simulator (again, 1000 times). This step is repeated until JIC has covered thirty break cases.

Figure 3 contains two graphs in which the independent variable is the number of break cases covered by JIC. In the left graph, 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. (This

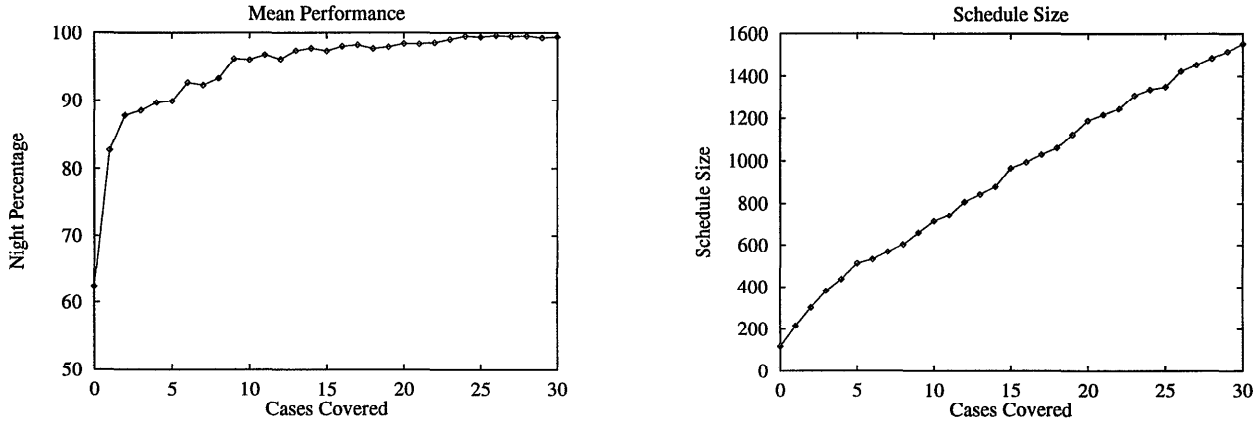

Figure 3: Mean performance, measured as night percentage, *vs.* cases covered and schedule size *vs.* cases covered.

indicates that JIC should be easily capable of finding extremely robust schedules in the hour before twilight.)

In the right graph of Figure 3, the dependent variable is schedule size, measured as the total number of actions the multiply contingent schedule contains. The nominal schedule contains 116 actions, and the results confirm, as expected, that schedule size increases linearly with the number of cases.

Figure 4 provides additional insight into the behavior of JIC. The graph shows the particular points at which a schedule (with a given number of cases covered) is likely to break throughout the night. It focuses on the first ten cases since this is where most of the improvement occurs. Each vertical line indicates the proportion of 1000 schedule executions that halted at a given percentage of the night, for a given number of cases. The night percentages are rounded to the nearest integer. The probability of executing the entire schedule is shown in the graph as the "break" probability at 100% of the night.

Consider the situation when JIC has not been run; *i.e.*, when zero break cases have been covered. There is a 0.4 probability of the nominal schedule breaking 14% of the way through the night, and there is a 0.4 probability of the nominal schedule executing through the entire night. After covering one case, JIC has managed to remove the early break at 14% and to increase the probability of executing completely through the night. However, by removing one possible break early on, JIC may introduce new possible breaks in the contingent schedule. For example, after one case is covered, a new possible break occurs at 25% of the night.

Figure 4 shows that schedule breaks tend to occur in only a few specific locations. There is a significant probability of the nominal schedule breaking early in the night. As JIC covers more cases, the probability of finishing the entire night increases. The probability of the break that is covered by an application of JIC is redistributed to the new contingent schedule; hence, the improvement gain is a function of the probability of successfully executing the new contingent schedule. The graph shows that JIC is able to cover many of the probable breaks, making the schedule extremely robust after only ten cases.

The results reported above are based on the duration uncertainty computed from a number of nights of actual telescope execution. How would JIC's performance be impacted by more or less uncertainty in the domain? To empirically investigate this question, we designed another experiment that uses the same scheduling problem as the experiment discussed above. The mean durations are also the same, but we experimentally vary the standard deviation for each action. Both JIC and the execution simulator use the same standard deviations. The standard deviation for each action is computed to be a given percentage of its mean; thus, a value of 1.0 for duration uncertainty indicates that the standard deviation for each action's duration is one percent of its mean.

Figure 5 shows the results from this experiment. The graph shows that when no cases have been covered by JIC, the mean percentage of night executed before breaking falls off quickly with increasing duration uncertainty. However, as more cases are covered by JIC, the rate of this fall-off decreases. Note that each successive line on the graph corresponds to a doubling of the number of cases covered; thus, the performance improvement expected from JIC is initially quite good, but it decreases with increasing cases (consistent with the results shown in Figure 3). For the actual standard deviations used in the previous experiment, we estimated the average percentage of the mean duration to be about 2.5. Figure 5 is consistent with our previous results: it predicts, for example, that for an uncertainty of 2.5 and 8 cases covered, the percentage of night executed should be around 95. This helps explain why JIC works so well on the actual telescope data. With greater duration uncertainty in the domain, we would not have been so fortunate!

Discussion and Conclusions

This paper has presented an algorithm for Just-In-Case scheduling. Using an existing 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 domain. (See Bresina, Drummond, & Swanson, 1994, for an evaluation of the practical impact of these modeling assumptions.) Traditional intuitions surrounding the management of stochastic actions suggest the inevitability of large search spaces and intractable reasoning. Using a "splitting" technique, our algorithm makes stochastic distinctions only when necessary. We have demonstrated in this paper that for a real problem, involving a large search space, there are very few stochastic action distinctions actually made by the algorithm. JIC covers most of the likely schedule breaks in a small number of iterations. In this concluding section, we discuss some related work and possible extensions.

The ideas behind JIC are quite general, and it should be possible to use the algorithm to manage other sorts of execution errors. All that is required is a statistical model of the frequency of execution breaks and a model of each break's impact on the state of the environment. For instance, in a machine shop one can gather statistics describing the mean time between failures for any given machine, and in a warehouse application involving human staff, typically there are numbers available that describe absenteeism. Such statistics could be put to use in a version of JIC. Computation time during idle periods (for instance, overnight) could be used to proactively reschedule for errors that occur during busy periods (for instance, during the day).

An interesting alternative for computing break probabilities is *stochastic simulation* (as used by Muscettola, 1993). Rather than explicitly propagate uncertainty intervals, such a technique generates a predicted duration for each scheduled activity according to a normal distribution. Working forward, stochastic simulation generates a specific start time for each scheduled action, producing one possible execution trace, or sample. A number of samples must be gathered to form a picture with any degree of confidence. In principle, this technique can be used to form more accurate break predictions than our simple uniform propagation model. However, it is not clear that the extra cost of the stochastic simulation technique is worth the extra accuracy it offers. This is a topic for future study.

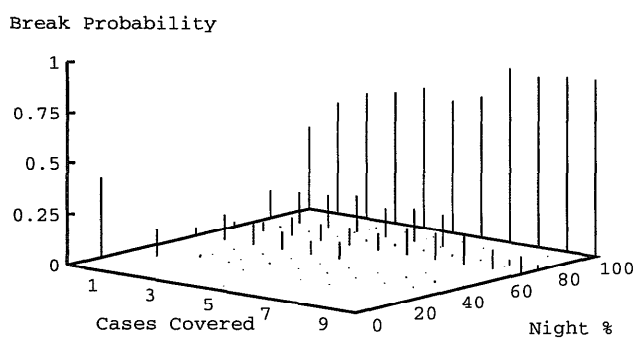

Figure 4: Where and how frequently the schedule breaks with respect to cases covered.

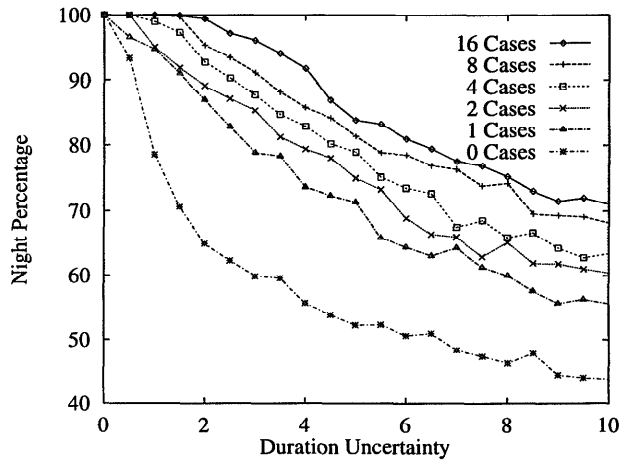

Figure 5: How performance of JIC falls-off with increasing duration uncertainty.

JIC is derived from an earlier algorithm, *traverse-and-robustify* (Drummond & Bresina, 1990), which requires an explicit model of stochastic action outcomes (Bresina, Drummond, & Kedar, 1993). The model used by traverse-and-robustify assumes that action outcomes are predefined discrete world states. In contrast, the actions considered in this paper are stochastic but continuous: the duration of an action can take on any value within some given interval. The traverse-and-robustify algorithm has been extended by Dean, *et al.* (1993) with the use of policy-iteration algorithms; however, their new algorithms still require an explicit and discrete stochastic action specification.

In contrast, Hanks (1990) presents an algorithm that forms its own stochastic action outcomes. Hanks' temporal projection system makes distinctions in the outcomes of an action only as required to answer a specific query, but it still assumes that actions have discrete outcomes. Our work can be viewed as a version of Hanks' idea that operates with continuous action outcomes. For our domain, it is necessary to make distinctions between possible action durations only when there is some chance that the next scheduled action will not be enabled. In essence, all actions are not created equal with respect to stochastic outcomes: temporal context is important and stochastic distinctions are introduced only when they matter.

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

First, there must be room for improvement. If the probability of successful execution of the nominal 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 because then each break case covered by JIC can usefully increase 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 nominal one. We plan to further investigate these intuitions, in order to more precisely characterize the conditions under which JIC works well.

Acknowledgments

Thanks to readers of previous drafts: John Allen, Will Edgington, Peter Friedland, Lise Getoor, Rich Levinson, Nicola Muscettola, and Pandu Nayak. Additional thanks to Will Edgington for assisting with the overall telescope management and scheduling project. Discussions on decision theory and contingent scheduling with Andrew Mayer, Othar Hansson, and Denise Draper have been very useful.

References

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 (IAPPP) Communications*, No. 52, T. Oswalt (ed).

Bresina, J., Drummond, M., and Kedar, S. 1993. Reactive, Integrated Systems Pose New Problems for Machine Learning. In *Machine Learning Methods for Planning*, S. Minton (ed.). Morgan-Kaufmann.

Bresina, J., Drummond, M., Swanson, K., and Edgington, W. *In Press.* 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.).

Bresina, J., Drummond, M., Swanson, K., and Edgington, W. 1993. Advanced Scheduling and Automation. *International Amateur Professional Photoelectric Photometry (IAPPP) Communications.*

Bresina, J., Drummond, M., Swanson, K. 1994. Managing Action Duration Uncertainty with Just-In-Case Scheduling. NASA Ames Technical Report FIA-94-04. (Also appears In *Working Notes of the 1994 AAAI Spring Symposium on Decision Theoretic Planning.* Stanford, CA.)

Dean, T., Kaelbling, L., Kirman, J., and Nicholson, A. 1993. Planning with Deadlines in Stochastic Domains. In *Proc. of AAAI-93*, pp 574 - 579.

Drummond, M., and Bresina, J. 1990. Anytime Synthetic Projection: Maximizing the Probability of Goal Satisfaction. In *Proc. of AAAI-90.* pp. 138–144.

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

Genet, D. 1994. AutoScope Control System Reference Manual (Software Version 2.0). AutoScope Corporation, Ft. Collins, CO.

Genet, R.M., and Hayes, D.S. 1989. *Robotic Observatories: A Handbook of Remote-Access Personal-Computer Astronomy.* AutoScope Corporation, Ft. Collins, CO.

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).

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

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

Muscettola, N. 1993. Scheduling by Iterative Partition of Bottleneck Conflicts, *Proc. of the 9th Conference on Artificial Intelligence for Applications*, IEEE Computer Society Press (March).

Swanson, K., Bresina, J., and Drummond, M. 1994. Just-In-Case Scheduling for Automatic Telescopes. In *Knowledge-Based Artificial Intelligence Systems in Aerospace and Industry.* W. Buntine & D.H. Fisher (eds.), Proc. SPIE 2244, pp. 10–19.

Thiebaux, S., Hertzberg, J., Shoaff, W., and Schneider, M. 1993. A Stochastic Model of Actions and Plans for Anytime Planning Under Uncertainty. In *Proc. of the Second European Workshop on Planning*, Vadstena, Sweden (Dec. 9–11).