Managing Action Duration Uncertainty with Just-In-Case Scheduling

John Bresina
Recom Technologies

Mark Drummond
Recom Technologies

Keith Swanson
NASA

AI Research Branch, Mail Stop: 269-2
NASA Ames Research Center, Moffett Field, CA 94035-1000
e-mail: {bresina, drummond, swanson}@ptolemy.arc.nasa.gov

Abstract

Some applications involve automatic generation and execution of schedules that contain actions with uncertain durations. Such uncertainty can cause schedules to break during execution. This paper presents a technique, called Just-In-Case scheduling or JIC, for building robust schedules that tend not to break. The technique 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 the scheduler to generate a contingent schedule for each highly probable break case, and produces a "multiply contingent" schedule. The technique was developed for a real telescope scheduling problem, and the paper presents empirical results showing that Just-In-Case scheduling performs extremely well for this problem.

1 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.

In modeling terms, an action with uncertain duration is *stochastic*: its outcome cannot be uniquely determined. It is traditionally believed that if a planning or scheduling problem contains stochastic actions, the resulting reasoning will be intractable. If the average stochastic branching factor is b then a sequential schedule containing l actions has approximately $l \times (b-1)$ different possible break points. Proactively managing these breaks would mean finding a schedule to cover each possible break point. The new schedule for a given break point would produce new possible break points that also require proactive management. Under these conditions, the number of possible break points grows too quickly, and the problem is intractable.

Counter to expectations about the difficulty of managing stochastic actions in domains of realistic size, the results presented in this paper demonstrate that reasoning about stochastic actions as implemented by JIC performs extremely well for our telescope scheduling application domain. We have reason to believe that it would work well for other similar domains. The application domain is explained in Section 2; the JIC algorithm is presented in Section 3 and empirically evaluated in Section 4; general discussion and conclusions are given in Section 5.

2 The Problem

JIC was developed for a real telescope scheduling problem. This section outlines only key aspects of the problem; more details are available elsewhere [2, 3, 7].

Telescope users electronically submit observation requests to a central location for subsequent scheduling. A request specifies both hard constraints and 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 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 [6]) – 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.

In this domain, action duration is a function of 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 statistics from

actual execution that characterize 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 cause the system to score poorly according to an objective function that penalizes wasted time. Our goal is to avoid schedule breaks without sacrificing schedule quality.

Schedules 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 on-line 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.

3 The Algorithm

JIC can be used with any scheduler that accepts a definition of a scheduling problem (here, a set of actions and constraints) and returns a schedule (here, a sequence of actions with enablement intervals). JIC interacts with a scheduler as follows. First, JIC analyzes the given nominal schedule and identifies the break with the highest estimated probability of occurrence. Then JIC forms a new scheduling subproblem for this break case and reinvokes the scheduler. JIC integrates the resulting schedule, as a contingent alternative, with the nominal one. Note that each contingent schedule can introduce new possible breaks, requiring further processing by JIC. In essence, JIC defines a "wrapper" algorithm that allows one to repeatedly invoke the existing scheduler to increase schedule robustness.

In our application, suppose that the 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 [6]. This leaves roughly fifty-nine minutes for JIC to make the schedule more robust. JIC incurs overhead to find the most probable 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.

For many applications there is more time available before execution can start than can be used by a given scheduler. While schedulers exist that can use all available time to produce ever-improving schedules, these are rather rare. Furthermore, these schedulers typically obtain most of the quality improvement in the first few scheduling iterations [13]. JIC can still be useful in conjunction with such schedulers since it could take advantage of any remaining available time.

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. We now consider each of these steps 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

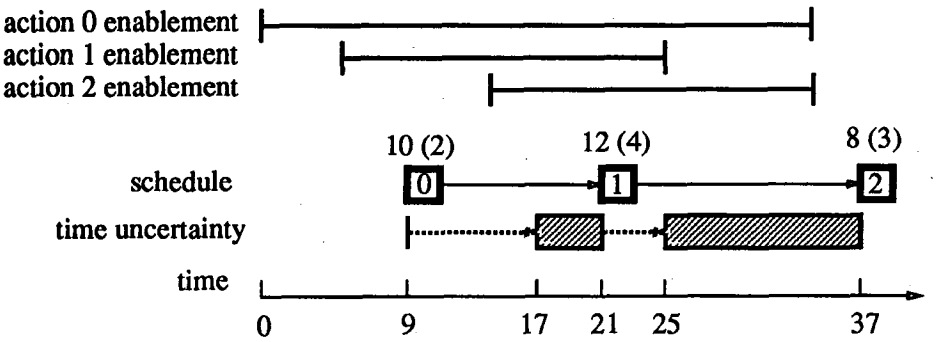

Figure 1: Propagation of temporal uncertainty.

uniform distribution in the current implementation of the JIC algorithm. The experimental results presented in the next section show that this approximation works quite well in practice.

In order to explain 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 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 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 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 implementation 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.

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. 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 these 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.

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

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.

$$\begin{aligned} &\text{For } i = 0\text{:} \quad p(\text{select}(A_i)) = 1.0. \\ &\text{For } i > 0\text{:} \quad p(\text{select}(A_i)) = p(\text{select}(A_{\beta(i)})) \times \\ &\qquad\qquad\qquad\qquad\quad p(\text{enable}(A_{\beta(i)})). \end{aligned}$$

Let $p(\text{break}(A_i))$ be the *break probability* for action A_i; that is, the probability that the schedule will break

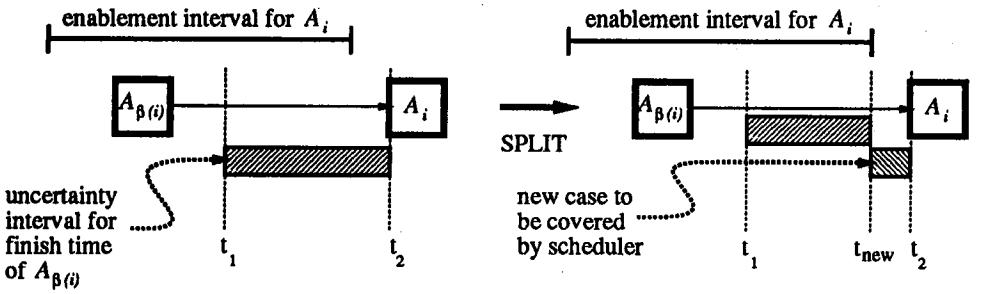

Figure 2: Splitting an uncertainty interval.

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 [12].

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.

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.

4 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 [8, 10]. The scheduler used in these experiments deterministically hill-climbs on a domain-specific heuristic [1]. 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 designed an experiment to measure how far into the night a multiply contingent schedule, with a given number of break cases covered, executes 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 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 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

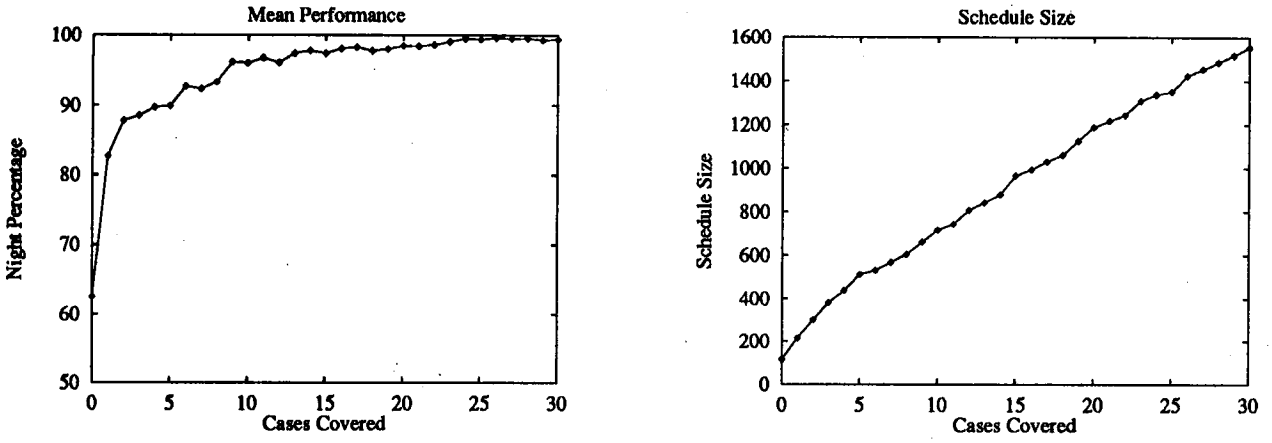

Figure 3: Mean performance, measured as night percentage, *vs.* cases covered and schedule size *vs.* cases covered.

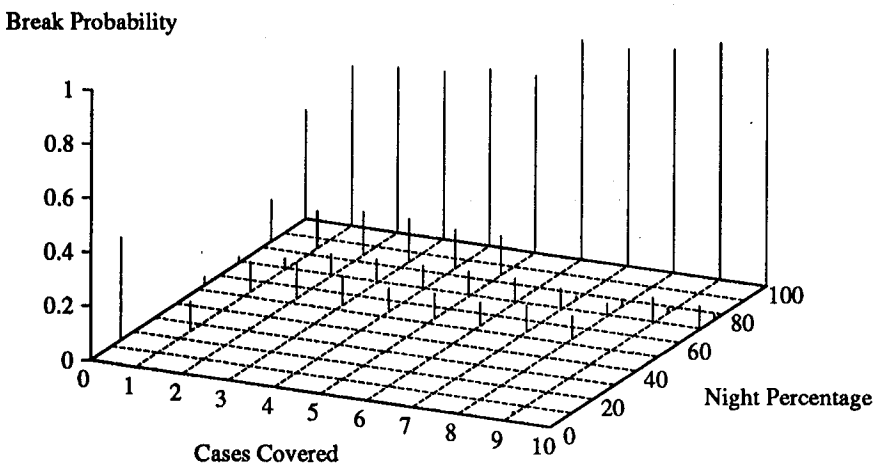

Figure 4: Where and how frequently the schedule breaks with respect to cases covered.

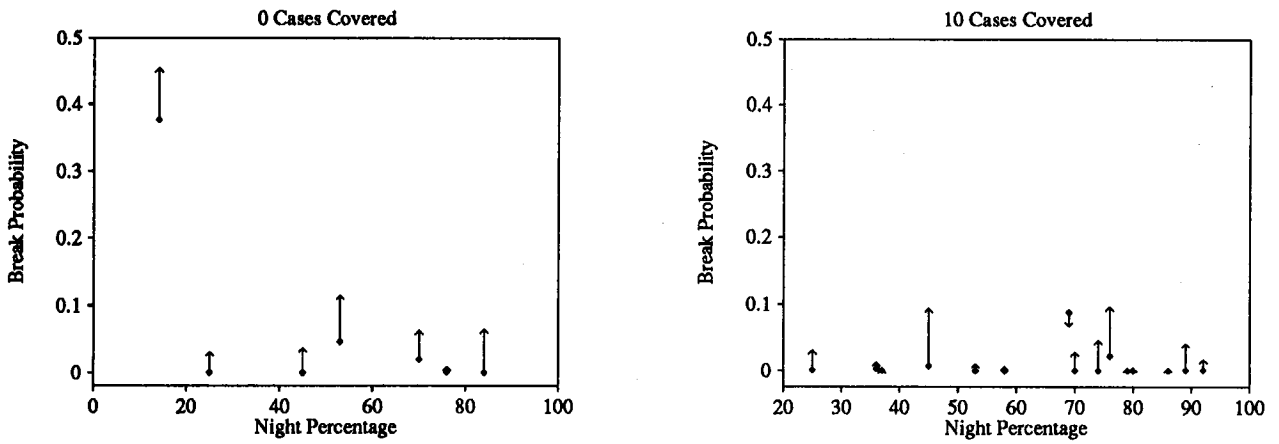

Figure 5: Comparison of predicted and actual break probabilities for the nominal schedule and after ten cases have been covered. The actual probability is shown as a point and a vector illustrates how the predicted probability differs from this point.

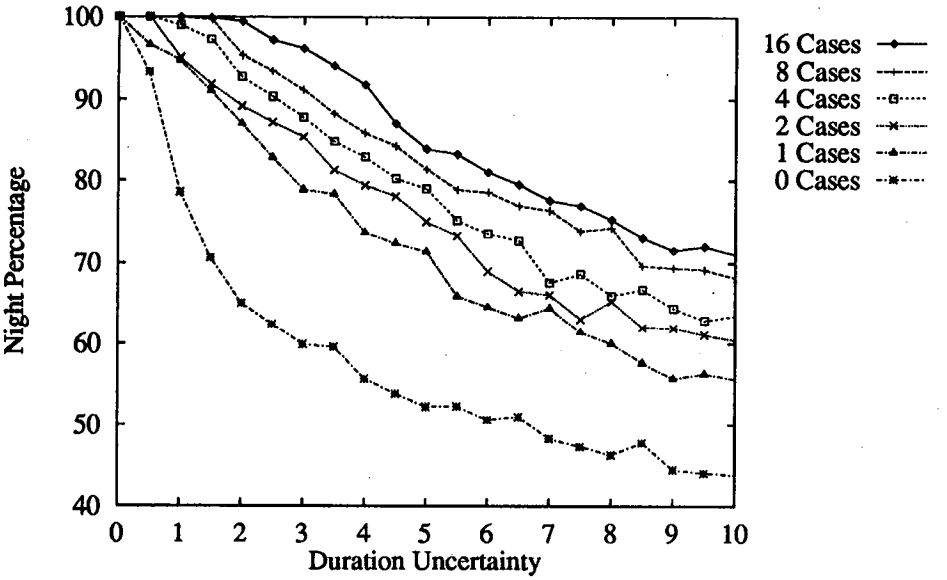

Figure 6: How performance of JIC falls-off with increasing duration uncertainty.

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 zeroth case, where JIC has not been run. 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 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 break occurs at 25% of the night.

Figure 4 shows that schedule breaks tend to occur in only a few specific locations. When no cases have been covered by JIC, there is a significant probability of the 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.

JIC's propagation of temporal uncertainty assumes that each action's duration and start interval have a uniform probability distribution. Thus, we would expect to see the predictions get worse along the length of the schedule. To evaluate the effect of these modeling assumptions, see Figure 5. The figure contains two graphs, both plot the difference between the break probabilities predicted by JIC and the actual break probabilities as experienced during execution.

The left graph shows the comparison for the nominal schedule, when no cases have been covered, and the right graph shows the comparison after ten cases have been covered. A data point indicates an actual execution break probability, and a vector anchored at that point indicates the magnitude and direction of the error in JIC's prediction. For instance, a vector dropping from a data point indicates that JIC predicted a lower probability for the break than actually occurred.

In the nominal schedule, JIC predicted the four breaks that actually occurred during execution and predicted three additional breaks that did not occur. After ten cases have been covered, JIC again predicts more break points than actually occur during execution. In both graphs, the magnitude of the prediction error at each break point is small – less than 10%. These results demonstrate that while the modeling in our current implementation is not precise, JIC's break probability predictions are reasonable.

The results reported above are based on the duration uncertainty computed from a number of actual observation nights. 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 6 shows the results from this experiment. The graph shows that when no cases have been covered by JIC, the 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 falls off 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 6 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!

5 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 problem. 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. Most of the likely schedule breaks are covered in a few iterations of JIC. In this concluding section, we discuss some related work and possible extensions to JIC.

An interesting alternative for computing break probabilities is *stochastic simulation* (as used by Muscettola [11]). Rather than explicitly propagating uncertainty intervals, such a technique can generate 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. It is not clear that the extra cost of the stochastic simulation technique is worth the potential extra precision it offers.

JIC is derived from an earlier technique, *Traverse-and-Robustify* [5], which required an explicit model of stochastic action outcomes. Furthermore, the stochastic outcomes were 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 technique was later extended by Dean, *et al.* [4] with the use of policy-iteration algorithms; however, their work also requires an explicit and discrete stochastic action specification.

In contrast, Hanks [9] 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 telescope scheduling domain, it is necessary to make distinctions in the duration of an action 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 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. We plan to further investigate these intuitions.

Another topic that requires further work is the connection between JIC and decision-theoretic approaches to scheduling. While not many schedulers are based on decision theory, there is a clear opportunity for this. Decision theory is concerned with the combination of reward and probability as expected utility. In a scheduling domain involving stochastic actions, a clear application of decision theory is in the area of attempting to maximize expected utility by finding a schedule that scores well according to an objective function and also has a high probability of being successfully executed.

While scheduling applications of decision theory are related to JIC, there are a couple of important distinctions. First, JIC is intended to address a space/time tradeoff: it uses off-line time and extra space to reduce the amount of time spent rescheduling on-line. This space/time issue is not explicitly addressed by decision theory. Second, the most obvious application of decision theory involves selecting the *single* sequence of actions that maximizes expected utility. The multiply contingent schedules that JIC forms include a number of alternative action sequences, conditioned on the current situation. The expected utility of a set of contingent schedules can be higher than any single schedule could achieve. In future work, we intend to investigate techniques that more tightly integrate decision theory and JIC, combining the use of expected utility in schedule synthesis with the robustness offered by multiply contingent schedules.

Acknowledgements

Thanks to readers of previous drafts: John Allen, Will Edgington, Lise Getoor, Rich Levinson, Nicola Muscettola, and Pandu Nayak. Additional thanks to Will Edgington for assisting on the telescope management and scheduling project.

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, T. Oswalt (ed).

[2] Bresina, J., Drummond, M., Swanson, K., and Edgington, W. 1993. 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.).

[3] Bresina, J., Drummond, M., Swanson, K., and Edgington, W. 1993. Advanced Scheduling and Automation. *International Amateur Professional Photoelectric Photometry Communications*, No. 52, T. Oswalt (ed).

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

[5] Drummond, M., and Bresina, J. 1990. Anytime Synthetic Projection: Maximizing the Probability of Goal Satisfaction. In *Proc. of AAAI-90.*

[6] 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.

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

[8] 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.

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

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

[11] 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).

[12] 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).

[13] Zweben, M., Daun, B., Davis, E., Deale, M. (*in press*). *Scheduling and Rescheduling with Iterative Repair.* In *Intelligent Scheduling*, M. Zweben & M. Fox (eds). Morgan-Kaufmann.