A more complex view of schedule uncertainty * based on contingency analysis

Anita Raja,†Victor Lesser, Thomas Wagner

UMass Computer Science Technical Report 1998-04

January 28, 1998

Abstract

The Design-to-Criteria scheduler is a domain-independent system that schedules complex AI problem solving tasks to meet real-time performance goals. In this paper, we further extend the scheduler to more effectively deal with uncertainty present in a schedule. This is based on an analysis of available schedules that can be used to recover from a situation in which partially executed schedules cannot be completed successfully. In addition to evaluating schedules effectively from the uncertainty perspective, we also implement method reordering techniques to minimize uncertainty.

* This material is based upon work supported by the National Science Foundation under Grant No. IRI-9523419, the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-97-1-0249, and Dept. of the Navy, Office of the Chief of Naval Research, under Grant No. N00014-95-1-1198. The content of the information does not necessarily reflect the position or the policy of the National Science Foundation, Defense Advanced Research Projects Agency (DARPA), Air Force Research Laboratory or the U.S. Government and no official endorsement should be inferred

†Computer Science Department, University of Massachusetts, Amherst, MA 01003, Email: araja@cs.umass.edu

A more complex view of schedule uncertainty based on contingency analysis *

Anita Raja† Victor Lesser Thomas Wagner

January 28, 1998

Abstract

The Design-to-Criteria scheduler is a domain independent system that schedules complex AI problem solving tasks to meet real-time performance goals. In this paper, we further extend the scheduler to more effectively deal with uncertainty present in a schedule that can be critical in hard deadline or hard cost situations. This is based on an analysis of available schedules that can be used to recover from a situation in which partially executed schedules cannot be completed successfully. In addition to evaluating schedules effectively from the uncertainty perspective, we also implement method reordering techniques to minimize uncertainty.

* This material is based upon work supported by the National Science Foundation under Grant No. IRI-9523419, the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-97-1-0249, and Dept. of the Navy, Office of the Chief of Naval Research, under Grant No. N00014-95-1-1198. The content of the information does not necessarily reflect the position or the policy of the National Science Foundation, Defense Advanced Research Projects Agency (DARPA), Air Force Research Laboratory or the U.S. Government and no official endorsement should be inferred

†Computer Science Department, University of Massachusetts, Amherst, MA 01003, Email: araja@cs.umass.edu

1 Introduction

The Design-to-Criteria scheduler [Wagner97c, Wagner98b] is a domain-independent system that schedules complex AI problem solving tasks to meet real-time performance goals. Problem solving tasks are modeled in the domain-independent TÆMS (Task Environment Modeling and Simulation) framework [Decker95,Decker93] that describes complex problem solving processes in terms of alternate ways by which problem solving goals can be achieved and the performance and resource requirements of these different approaches. A simplified example of a TÆMS task structure for searching the Web for information on reviews on Adobe Photoshop is shown in Figure 1. The scheduler determines a particular path to achieve a goal as well as the specific order of execution of the subtasks associated with this path. It uses a complex user-defined scheduling criteria [Wagner97c] that takes into account the performance characteristics such as cost, quality and duration in the overall schedule and amount of uncertainty with respect to these characteristics.

In this paper, we further extend the scheduler to more efficiently deal with uncertainty present in a schedule. This is based on an analysis of available schedules that can be used to recover from a situation in which partially executed schedules cannot be completed successfully. In addition to evaluating schedules more effectively from the uncertainty perspective, we also implement method reordering techniques to minimize uncertainty. The Design-to-Criteria scheduler with its present functionality does some reordering of subtasks within a schedule [Wagner97e] but it does not reason about whether there are ways to recover from failure scenarios.

We define *schedule robustness* as a characteristic of a schedule in which the schedule allows for recovery from execution failure of one of the scheduled actions. In evaluating a schedule, we want to take into account whether there exist alternative ways of completing the schedule, i.e., achieving the high level task, if the schedule should fail during the course of execution. This type of analysis, called *contingency planning* can be expensive because it could involve an exhaustive search for the appropriate method that would improve to schedule robustness without diminishing the criteria requirements [Bresina94]. However, the technique we describe in this paper implements an algorithm which eliminates the need to do an exhaustive search, though it is more expensive than our non-contingency scheduling approach.

In this paper, we discuss contingency scheduling issues and formalize them using five statistical measures of *schedule robustness*. We then present a computationally feasible algorithm for building robust schedules and demonstrate their efficiency via experimental results.

2 The Expected Lower Bound and the Approximate Expected Bound

In this section we try to answer the following questions:

1. *How can we effectively predict the performance of a schedule when there is uncertainty in the performance of methods in the schedule?*
2. *What are the different approximations to this value and when is a specific approximation appropriate?*

2.1 Expected Lower Bound Rating

We describe a simple example which in a concise manner captures the intricacy and functionality of contingency analysis.

TÆMS models are comprised of *tasks*, *methods*, which are executable tasks or primitives, and *non-local-effects* (NLEs). Tasks are non-executable methods which are decomposed into subtasks, methods, or both. TÆMS methods are described statistically via discrete probability distributions in three dimensions: quality, cost and duration. Quality in the given figure describes the contribution of a particular action to the top level task. Duration describes the amount of time a method will take to execute and Cost describes the financial or opportunity cost inherent in performing the action modeled by the method.

The oval nodes in Figure 1 are tasks and the rectangular nodes are methods. The top level task is Find-review-Information-on-Adobe-Photoshop. This high level task can be achieved by either completing task Query-Benchin-Site(A) successfully or executing the method Search-Adobe-URL(B) or both. If both A and B are executed the maximum quality is taken. This is described in TÆMS by means of the max() *quality accumulation function* (qaf).

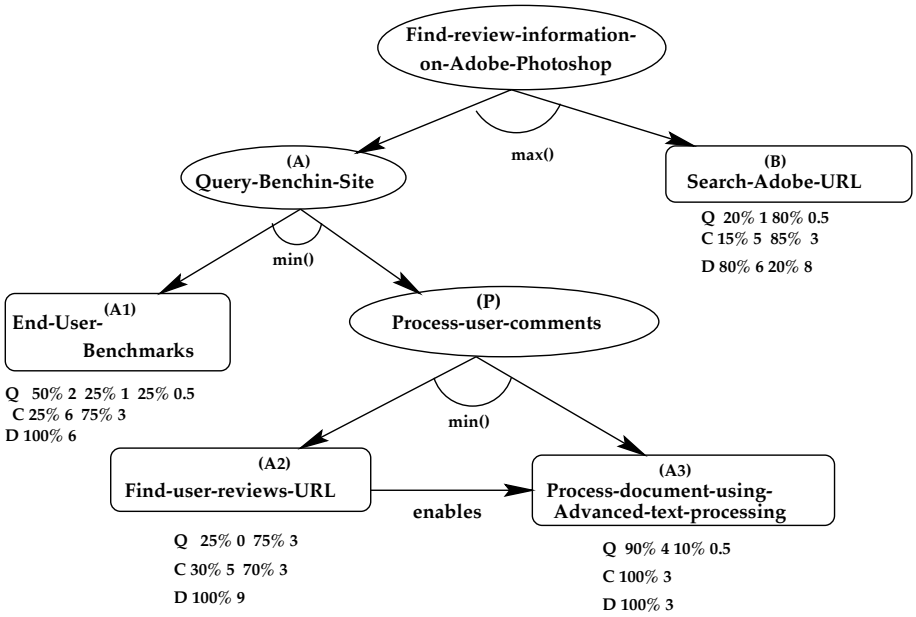


Figure 1: Gather review information on Adobe Photoshop.

Task Query-Benchin-Site has access to Benchin's end-user review synopsis and also entails processing comments by other users on the product. We assume for the purposes of this scheduling problem, that the product reviews by Benchin alone are inadequate data and have to be supplemented by user-reviews to satisfy client requirements. This relationship between End-User-Benchmarks(A1) and Process-user-comments(A2) is described by means of the min() qaf and the minimum quality value from the sub methods is chosen. This forces the scheduler to schedule both tasks to avoid a 0 quality being propagated by the min().

The *enables* NLE between Find-user-reviews-URL(A2) and Process-document-using-Advanced-processing(A3) indicates that Find-user-reviews-URL needs to incur a non-zero quality for Process-document-using-Advanced-processing to be executed.A *facilitates* NLE, used later in the paper, describes a soft relationship between methods, where a non-zero quality achieved by the facilitator allows the expected performance of the facilitatee to improve by the degree of facilitation.

The quality of results achieved by executing task Query-Benchin-Site is very high and hence preferred. However it is possible that for certain products,

Benchin does not contain user reviews and we suppose this happens 25% of the time. In this case, we would prefer method Search-Adobe-URL (which provides the company's marketing description of the product which though biased in complementing the product, still provides relevant information on the capability of the product) and has a 100% guarantee of achieving the top-level goal even if it is of lower quality.

Lets assume the criteria requirements are that the task should achieve maximum quality with a duration deadline of 18 minutes. The Design-to-Criteria scheduler first enumerates a subset of the *alternatives* that could achieve the high level task. An alternative is an easy to compute schedule approximation with an estimate for quality, cost and duration distributions that will result from scheduling the alternative. A subset of these alternatives are selected and schedules are created using a heuristic single-pass method-ordering technique. The set of candidate schedules are then ranked using a sophisticated multi-dimensional evaluation mechanism [Wagner97c] which compares the schedules' statistical attributes to scheduling design criteria, e.g., quality, cost, duration and uncertainty measures, provided by scheduler clients.

For the remainder of this document, we will modify and simplify the scheduler's criteria-driven evaluation mechanism to make examples and comparisons succinct; that is we will focus only on the expected quality attributes of schedules and ignore the multi-dimensional and relative scaling components of the scheduler's standard utility calculation. The term *rating* will thus denote the expected quality of a given schedule and nothing more. Comparisons between the schedules produced by the contingency mechanisms and the standard Design-to-Criteria scheduler are still valid because this simplification is reflected in the goal criteria submitted to the scheduler. All scheduling in this paper will work to maximize quality within a given hard deadline.

In this paper, we will call the simplified quality-only rating returned by the standard Design-to-Criteria scheduler the *expected lower bound* and view it as the statistical measure of the characteristics of a schedule assuming no rescheduling.

For our first example, we will focus on maximizing quality within a hard deadline of 18 minutes. The two schedules we will examine are: {A1,A2,A3} and {B}. The following are the Expected Lower Bound (ELB) ratings for

these schedules:[1]

1. {A1,A2,A3}: Rating: 0.72 (Expected Quality)
   Quality : (44% 0.0) (18% 0.5) (25% 2.0) (13% 1.0)
   Cost: (52% 9.0) (23% 11.0) (17% 12.0) (8% 14.0)
   Duration : (100% 18)
2. {B}: Rating 0.6 (Expected Quality)
   Quality : (20% 1) (80% 0.5)
   Cost: (15% 5) (85% 3)
   Duration: (80% 6) (20% 8)

The schedule {A1,A2,A3} is chosen and executed since it has the best expected lower bound rating. A1 executes successfully , then A2 executes and suppose A2 fails (i.e. it results in 0 quality), which happens 25% of the time. Then A3 fails to get enabled and the schedule breaks since there is no time left to reschedule {B} as an alternate schedule.

Because of the one-pass low-order polynomial method sequencing approach used by the scheduler to control scheduling combinatorics, the standard Design-to-Criteria scheduler will only produce one permutation of the methods A1, A2, and A3. However, if the scheduler did produce multiple permutations, the schedules {A1,A2,A3} and {A2,A1,A3} would receive the same expected lower bound value. Hence the contention is that there is no difference in performance if either of the two was chosen, or produced by the method ordering heuristics. However on more detailed evaluation of the schedules, we see that {A2,A1,A3} allows for recovery and contingency scheduling which schedule {A1,A2,A3} does not permit (Figure 2) for the given deadline. If {A2,A1,A3} is the schedule being executed and A2 fails, there is time to schedule method {B} and complete task TG1. This clearly implies that schedule {A2,A1,A3} should have a better expected performance rating than {A1,A2,A3} as the schedule {A2,A1,A3} includes the recovery option from failure in its structure.

[1]In this particular example, the ELB is an underestimate of expected schedule quality as the calculation is not based on a contingency-tree style analysis due to the combinatorics of the general Design-to-Criteria scheduling problem. Though this example is amenable to that type of calculation, the general case is not. We are considering using limited tree expansion in situations such as these where it is not precluded by the combinatorics of the actual scheduling instance.

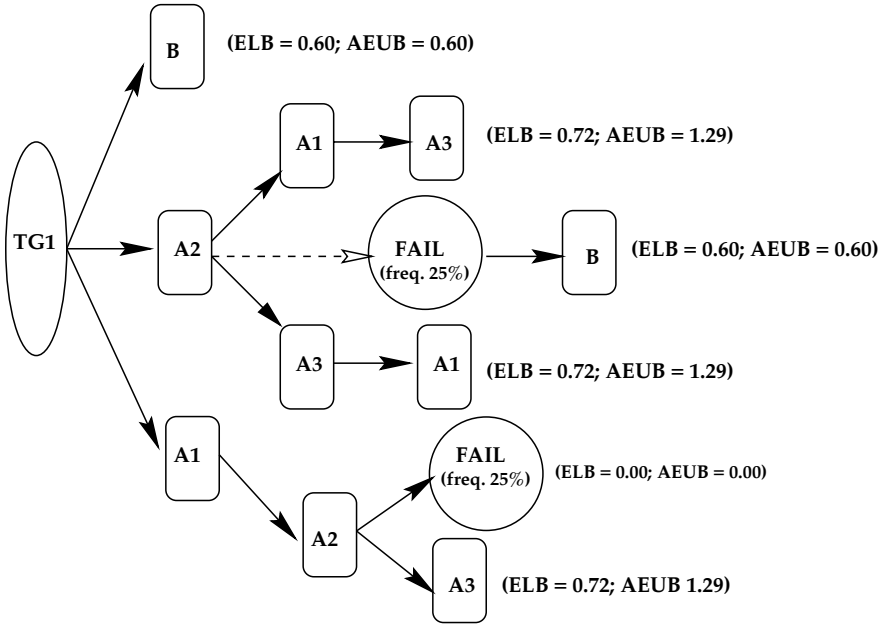


Figure 2: Schedule options for example one with (schedule rating) values

2.2 Critical Regions

In our example, task A2 has an enables non-local effect [Wagner97c] as well as a 25% chance of failure within its distribution. We hence predict that task A2 could potentially be a *critical region*. A critical region is a set of possible outcomes of method execution which if occurred would seriously degrade the performance characteristics of the overall schedule. In order to understand the implications of this potential critical region, let us remove the failure possibility from the performance characterization of A2 and replace method A2's 25% chance of quality 0 by the expected value of the distribution. Method A2 hence is assigned a quality of 3, with a probability of 1. The Design-to-Criteria scheduler is reinvoked with the modified task structure and reschedule. The following are the ratings returned by the scheduler.

1. {A1,A2,A3}: Rating 1.29 (Expected Quality)
   Quality : (32% 0.5)(22% 1.0)(45% 2.0)
   Cost : (52% 9.0) (23% 11.0) (17% 12.0) (8% 14.0)
   Duration: (100% 18)
2. {B} : Rating 0.6 (Expected Quality)
   Quality: (20% 1) (80% 0.5)

Cost: (15% 5) (85% 3)
Duration: (80% 6) (20% 8)

The performance measure for the modified task structure is no longer the expected lower bound, instead it is the approximate upper bound as it describes the expectations if failure is not possible. The schedule {A1,A2,A3} now receives a rating of 1.29. The $\frac{1.29-0.72}{0.72} * 100 = 79\%$ improvement in quality with respect to the expected lower bound rating is significant. This 79% improvement in performance measure confirms that the possibility of failure of method A2 significantly decreases the rating of schedule {A1,A2,A3}. So now we consider the optional schedules for the original task structure to neutralize the effect of this critical region.

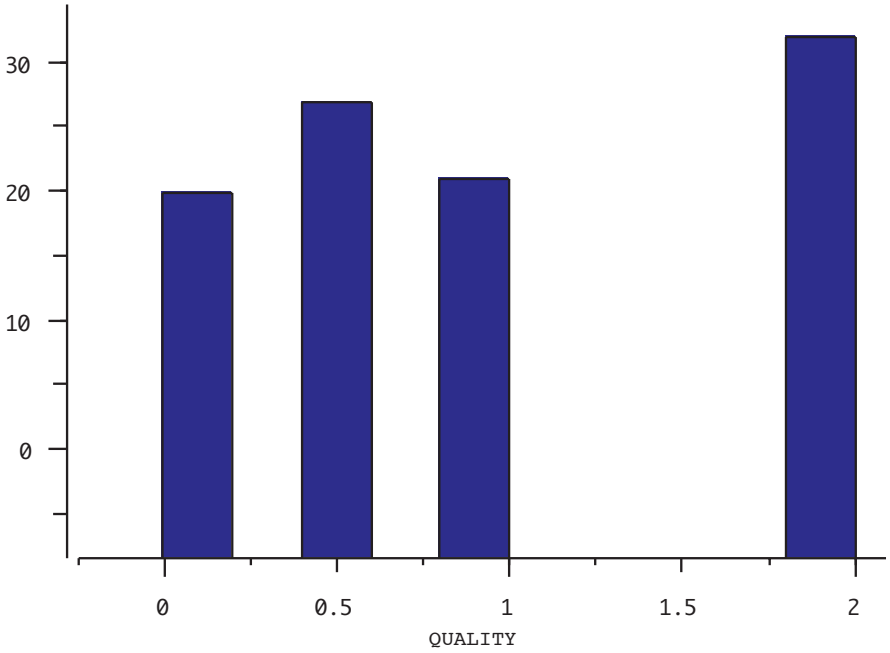
The tree structure in Figure 2 presents all the options of schedule generation that will meet the criteria. From this diagram, we see that schedule {A1,A2,A3} does not have an option to reschedule and still meet the deadline, if method A2 produces an undesirable outcome.

So we consider a simple reordering of schedule {A1,A2,A3} which is {A2,A1,A3}. To assess the effects of rescheduling when A2 fails on this schedule {A2,A1,A3}, we combine the ratings for schedules $\{A2^{success}, A1, A3\}$ and $\{A2^{failure}, B\}$ based on their likelihoods of occurrence. So a schedule starting with A2 gets a rating of $\frac{75}{100} * 1.29 + \frac{25}{100} * 0.60 = 1.1175$. We use a similar analysis to get the values of schedules starting with A1 $= \frac{75}{100} * 1.29 + \frac{25}{100} * 0 = 0.9675$and B$= 1 * 0.60 = 0.60$

This type of evaluation of the schedule is what we call the Approximate Expected Bound(AEB), which will be formally defined in the next section.

So schedule {A2,A1,A3} has a better performance guarantee than {A1,A2,A3}. The ELB computation of the Design-to-Criteria scheduler evaluates the performance measure of both {A1,A2,A3} and {A2,A1,A3} to be the same as it does not take into account the recovery options present within {A2,A1,A3} while evaluating it. This leads us to believe that the ELB perhaps is not the most appropriate performance measure for all task structures, particularly where hard deadlines or cost limits (in contrast to soft preferences) are important.

In Figure 3, we show the statistical performance of the schedule with the highest ELB {A1,A2,A3} for a 100 simulation runs. We note that the sched-

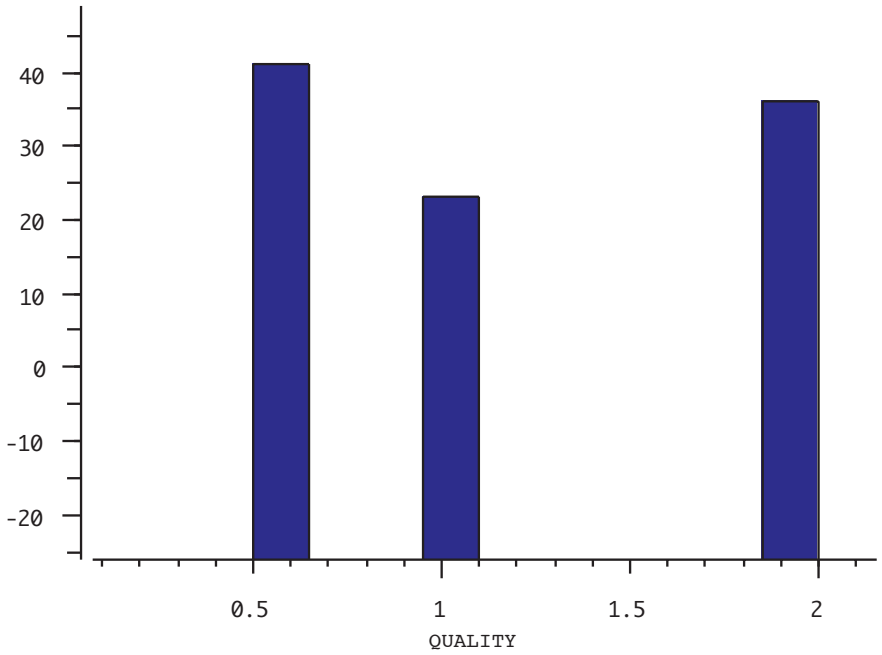


Histogram OF Quality[Simulation-Using-Elb-Measure]

Figure 3:

ule fails to achieve any quality about 20% of the time. The mean quality achieved by using this performance measure is 0.98.

Figure 4 describes the statistical performance data for the schedule with the highest AEB {A2,A1,A3} over 100 simulation runs. The schedule always achieves a non-zero quality value due to the built-in contingency and the mean quality achieved here is 1.96. Also note that the AEB is a better estimator of actual performance than the ELB.



Histogram OF Quality[Simulation-Using-Aeb-Measure]

Figure 4:

3 Performance Measures

In this section we try to formalize a general theory relating to the concepts on contingency discussed in our previous example. The question we strive to answer formally here is the following: *What is a good estimator of the actual execution behavior of a schedule and under what scenarios?*

Our basic approach is to analyze the uncertainty in the set of candidate

schedules to understand whether a better schedule can be selected or an existing schedule can be slightly modified such that its statistical performance profile would be better than that normally chosen by the Design-to-Criteria scheduler.

Some basic definitions are given below:

1. A schedule s is defined as a set of methods $m_1, m_2, ...m_{n-1}, m_n$.
2. Each method has multiple possible outcomes, denoted m_{ij}, where j denotes the j'th outcome of method m_i.
3. Each outcome is characterized in terms of quality, cost, and duration, via a discrete probability distribution for each of these dimensions.
4. m_{ij}^{cr} is a critical region when the execution of m_i results in outcome j which has a value or set of values characterized by a high likelihood that the schedule as a whole will not meet its performance objectives.
5. A schedule s_{ij}^{cr} is called a critical path if it is defined as $m_1 ...., m_{i-1}, m_{ij}^{cr}, m_{i+1}, ...m_{n-1}, m_n$. The performance characteristics of s_{ij}^{cr} are not likely to meet successful overall performance criteria desired for the schedule.
6. f_{ij}^{cr}, the frequency of occurrence of a path s_{ij}^{cr}, is defined as the probability of the path s_{ij}^{cr} being executed with the associated outcomes of a specific method(i.e. m_{ij}^{cr}).
7. $\overline{m_{ij}^{cr}}$ is m_{ij}^{cr} with its current distribution being redistributed and normalized after the removal of its critical outcome. In other words, the criticality of m_{ij}^{cr} is removed and the new distribution is called $\overline{m_{ij}^{cr}}$.
8. $\overline{s_{ij}^{cr}}$ is the schedule $m_1 .., m_{i-1}, \overline{m_{ij}^{cr}}, m_{i+1}, ...m_{n-1}, m_n$.

We describe five statistical measures for a specific single schedule:

1. Expected Lower Bound(ELB)

The expected lower bound rating, of a schedule s_{ij}, is the performance measure of a schedule execution without taking rescheduling into consideration [Wagner97c]. It is a expected rating because it is computed on a statistical basis taking quality, cost and duration distributions into account.

2. Approximate Expected Upper Bound(AEUB)

It is the statistical schedule rating after eliminating all regions where rescheduling could occur. The assumption is that there are no failure regions and hence the schedule will proceed without any failures and hence no rescheduling will be necessary. The following is a formal definition of AEUB:

Suppose m_{ij}^{cr} is a region in the schedule $s = \{m_1..m_n\}$ and it occurs with frequency f_{ij}^{cr}. Let $s_{ij}^{\overline{cr}} = \{m_1, m_2..\overline{m_{ij}^{cr}}..m_n\}$.

If $\frac{ELB(\overline{s_{ij}^{cr}})-ELB(s)}{ELB(s)} \geq \alpha$, then m_{ij} is a critical region. α is a domain dependent measure giving an upper bound for the improvement in the schedule performance prediction.

For example 1, we see that

$\frac{ELB(\{\overline{A2},A1,A3\})-ELB(\{A1,A2,A3\})}{ELB(\{A2,A1,A3\})} \geq 0.8$. Hence there is at least an 80% increase in the schedule rating if the likelihood of failure of A2 is removed.

When this computation is done on an entire schedule for all of its critical regions, we call it the Approximate Expected Upper Bound.

Generalizing this formula for k critical regions $m_{i_1j_1}...m_{i_kj_k}$,

$AEUB(s) = ELB(\{m_1...m_{i_1-1}, \overline{m_{i_1j_1}^{cr}}..\overline{m_{i_2j_2}^{cr}}.....\overline{m_{i_kj_k}^{cr}}...m_n\}$.

The AEUB is thus the best rating of a schedule on an expected value basis without any rescheduling.

3. Optimal Expected Bound (OEB)

It is the schedule rating if rescheduling were to take place after each method execution. So the first method is executed, a new scheduling subproblem which includes the effects of the method completion is constructed and the scheduler is re-invoked. The first method in this new schedule is executed and the steps described above are repeated. Hence the optimal schedule is chosen at each rescheduling region. For complex task structures, the calculation would require a tremendous amount of computational power and is unrealistic to use in measuring schedule performance in a real system.

In most situations, $ELB(s) \leq OEB(s) \leq AEUB(s)$, since the $OEB(s)$ is based on recovery from a failure while $AEUB(s)$ assumes no failure.

4. Expected Bound (EB)

Let m_i^e be the set of values for the actual outcome class when method m_i is executed. After each method execution the schedule is re-rated. If for some m_i^e,

$ELB(\{m_1, m_2...m_n\}) \gg ELB(\{m_1^e, m_2^e, m_3^e...m_i^e, m_{i+1}..m_n\})$, then a new schedule is constructed based on the partially complete schedule $\{m_1^e, m_2^e, ...m_i^e\}$.

So the EB is the schedule rating when rescheduling occurs only when there is a possibility for the partial execution of the current schedule will fail to meet expected criteria as a result of the outcomes of methods already executed. This computation, like the OEB, will require extensive computational power. Again in most situations, $ELB(s) \leq EB(s) \leq OEB(s) \leq AEUB(s)$.

5. Approximate Expected Bound(AEB)

It is the schedule rating with rescheduling only at critical regions and using expected lower bound of the new stable schedule for methods following the critical region. This is limited contingency analysis at critical regions.

Consider a schedule s of n methods $\{m_1, m_2..m_i..m_n\}$. Now suppose m_{ij} is a critical region with a frequency of occurrence of f_{ij}. In order to compute the AEB of the schedule, we replace the portion of the schedule succeeding m_{ij}^{cr}, which is $\{m_{i+1}, m_{i+2}, ...m_n\}$ by $\{l_{i+1}, l_{i+2}.....l_k\}$ if there exists a $\{l_{i+1}, l_{i+2}.....l_k\}$ such that $ELB(\{m_1...m_{ij}^{cr}, l_{i+1}...l_k\}) \geq ELB(\{m_1...\overline{m_{ij}^{cr}}, m_{i+1}...m_n\})$.

The Approximate Expected Bound for this instance is computed as follows: $AEB_{ij}(\{m_1, ...m_n\}) = ELB(\{m_1...\overline{m_{ij}^{cr}}, m_{i+1}..m_n\}) * (1 - f_{ij}) + ELB(\{m_1..m_{ij}^{cr}, l_{i+1}..l_k\}) * f_{ij}$.

The new schedule rating thus includes the rating from the original part of the schedule as well the ELB of the new portion of the schedule. This is basically the calculation at the end of Section 2.

Now we describe the general case scenario. Let {m_1,m_2,m_3,...m_i,...m_n} be a schedule s of n methods with k critical regions named $m^{cr}_{i_1j_1}, m^{cr}_{i_2j_2}.....m^{cr}_{i_kj_k}$. Let the recovery path available at each critical region m^{cr}_{ij} be s^r_{ij} and each m^{cr}_{ij} occurs with frequency f^{cr}_i. The AEB of the entire schedule is described recursively as

$$AEB = ELB(\{m_1...m^{cr}_{ij}, l_1, ...l_k\}) * f^{cr}_i + AEB(\{m_1....\overline{m^{cr}_{ij}}, m_{i+1}, ...m_n\}) * (1 - f^{cr}_i)$$

which can be expanded out as follows:

$$AEB = f^{cr}_1 * ELB(\{m_1.....m_{i_1-1}, m^{cr}_{i_1j_1}, l_{a1}.....l_{b1}\})+$$

$$(1 - f^{cr}_1) * f^{cr}_2 * ELB(\{m_1...\overline{m^{cr}_{i_1j_1}}...m^{cr}_{i_2j_2}, l_{a2}.....l_{b2}\})+....$$

$$(1-f^{cr}_1)*..*(1-f^{cr}_{k-1})*f^{cr}_k*ELB(\{m_1..\overline{m^{cr}_{i_1j_1}}..\overline{m^{cr}_{i_2j_2}}..\overline{m^{cr}_{i_3j_3}}..m^{cr}_{i_kj_k}, l_{ak}..l_{bk}\})+$$

$$(1 - f^{cr}_1) * (1 - f^{cr}_2) * .. * (1 - f^{cr}_k) * \underbrace{ELB(\{m_1..\overline{m^{cr}_{i_1j_1}}..\overline{m^{cr}_{i_2j_2}}..\overline{m^{cr}_{i_kj_k}}..m_n\})}_{AEUB}$$

The above computation produces an approximate measure since we use the $ELB(\{m_1..m_{ij}, l_{i+1}..l_k\})$. A better and more exact computation would be to use the $AEB(\{m_1..m_{ij}, l_{i+1}..l_k\})$. So if we recursively refine the $ELB(\{m_1..m_{ij}, l_{i+1}, ..l_k\})$, the schedule rating approaches the expected bound (EB). Thus, the deeper the recursion in the analysis of critical regions, the better the schedule performance measure and the closer it is to the actual performance measure when rescheduling occurs. This describes the anytime nature of the AEB computation. Thus, in most situations, $EB(s) \geq AEB(s)$ and the $AEB(s) \geq ELB(s)$ by definition.

Here we would like to add that all computations above are based on heuristics and hence are approximations including the OEB and EB. We could define AEUB',OEB',EB', AEB' and ELB' which would involve complete analysis of all paths by the scheduler. The resulting schedules would display higher performance characteristics and meet goal criteria better but will also be computationally infeasible to generate [Wagner98b].

4 Rescheduling and Recovery Algorithms

In this section, we describe a generic algorithm which can guarantee a more precise performance evaluation of schedules when uncertainty is present in the schedule, using the theory described above.

Algorithm for building stable schedules

The following is a formal description of the algorithm which chooses the schedule that provides the best performance guarantee statistically : Let s^b = {m_1,m_2,m_3,...m_i,...m_n} be the best schedule returned by the Design-to-Criteria scheduler for a given task structure. Suppose the scheduler evaluates k schedules to decide which is the best schedule, where $s_k = \{m^k_{i_1}...m^k_{i_n}\}$ and let S be the set of all k schedules. s^b has the highest ELB in S. Let $S_{rem} = S - \{s^b\}$. Then $ELB(s^b) \geq ELB(s_{rem})$ for all $s_{rem} \in S_{rem}$.

Let S^b_{rem} be the set of $s_{rem} \in S_{rem} \ni AEUB(s_{rem}) \gg ELB(s^b)$. If $S^b_{rem} \neq \phi$, then we compute the $AEB(s)$ for each $s \in \{S^b_{rem} \bigcup s^b\}$. The new best schedule s^b_{aeb} is the one with with the highest AEB.

s^b_{aeb} is guaranteed be more robust and also the AEB rating s^b_{aeb} is a better performance estimate of the actual performance of the schedule.

Identifying critical regions

The AEB is a better estimate than the ELB when there is uncertainty in the schedule, i.e., there are critical regions in the schedule and there is a possibility for contingency plans. This could relate to any of the following factors:

1. Significant variance in the criteria distribution: For methods with a single outcome, we look for variance in the criteria distribution of the method from the expected values and evaluate if this variance will critically affect the performance of the schedule. In our example, method A has a 25% chance of failure. This makes it a candidate critical point.

2. Significant likelihood of failure: For methods with multiple outcomes, we determine if the other outcomes which are not included in the schedule could detrimentally affect the schedule's performance if they occurred. We also examine the distributions of methods whose performance could affect other methods as described by non-local effects, namely the enablers and facilitators in a task structure. In the example, method A2 also enables method A3. Thus, it becomes imperative to evaluate A2 as a critical point.
3. Reasonable deadline: The AEB calculation is useful when there is a rigid deadline allowing enough time for contingency but not for redundancy. If cost is not an issue and the duration deadline for a task structure is elastic enough for the scheduler to schedule(using the ELB measure) redundant activities to overcome critical regions, then contingency analysis might not be required. However, we would like to point out that while the schedule with highest ELB rating would execute the redundant method(s) regardless of the success or failure of the critical region, the schedule with the highest AEB can dynamically adjust to the actual execution outcomes and hence execute the method(s) which will best improve performance with minimal redundancy and cost.

   In our example, the duration deadline of 18 minutes allows for contingency but not for redundancy. However with a duration deadline of 30 minutes, the ELB computation produced the schedule {A1,A2,A3} as there is enough time to reschedule B in case of failure of A2. The AEB computation also chose the schedule {A2,A1,A3} since both duration and cost are not constrained. Both schedules had the same performance statistically with a mean quality of 1.09 as expected.

We have heuristics which allow us to perform cheap approximate analysis of the task structure and schedule to analyze the existence and effects of critical points. This helps determine whether contingency analysis is possible and worth the effort.

Method reordering

Earlier, we noted that the AEB evaluation, unlike the ELB evaluation, views permutations of the same set of methods as different schedules. We saw that

while one permutation A2,A1,A3 permitted a contingent schedule, the other A1,A2,A3 did not. We describe below two types of method reordering within a schedule:

Simple reordering: Consider a schedule $s = \{m_1,m_2,m_3,...m_i,...m_n\}$. Suppose m_i is a critical point. Then if the AEB computation is unable to find a contingent schedule in case of failure of m_i , we will automatically try to move m_i ahead in the schedule without affecting any of the non-local effects such as enables or facilitates. So if m_i can be moved ahead of m_3 without affecting any non-local effects, we get a new schedule $s\prime = \{m_1,m_2,m_i,m_3,.....\}$ and we reevaluate the AEB rating. Our example uses simple reordering i.e. A2 can be moved ahead of A1 and a contingent schedule can be obtained.

Complex reordering: Consider the schedule s again but suppose m_{i-1} facilitates m_i, which is a critical point. Also suppose we are unable to find a contingent schedule in case m_i fails. Here, we would try to move method m_i forward in the schedule, by ignoring the facilitates and evaluate if the AEB rating of the new schedule justifies the loss of the facilitates.

5 Conclusions

This paper has presented an algorithm to improve the performance of a schedule. Using the schedules emitted by the Design-to-Criteria scheduler and statistical measures of schedule evaluations, the algorithm builds contingent schedules to improve overall robustness. In the examples discussed above, we assume distinct failure regions, the instance being failure to comply with the duration deadline. This requirement simplifies the examples and the explanation – the work presented here is extendible to other hard performance criteria as well.

[Bresina94] discusses an algorithm for a specific domain namely a real telescope scheduling problem where the stochastic actions are managed by a splitting technique. Here the Just-In-Case scheduler pro-actively manages duration uncertainty by using the contingent schedules built as a result of analyzing the problem using off-line computations.

Our work differs from previous work done in construction of contingent schedules as our analysis is done in interactive time even as the problem

is being solved and hence we have real duration and cost constraints in evaluating the entire search space. Secondly, we are constantly evaluating the user specifications with the criteria constraints to get a satisficing yet robust result. Also our algorithm takes advantage of the structural analysis of the problem, namely the TÆMS task structure representation, to reduce the complexity of the search problem.

We still use approximations and statistical measures of schedule criteria values and hence cannot guarantee 100% reliable schedules for all problems within our domain. The tradeoff between robust schedules and criteria constraints is not the same for all users or for all problems within a domain. And so our approach is a step towards guaranteeing robustness where there are some resources set apart for this contingency analysis.

In this paper, we define critical regions as those methods or tasks in the schedule which could potentially cause the schedule to break. We have identified four different types of critical regions.

1. Structural critical regions: These are regions where failure is obvious, for instance zero quality under a min, a enables relationship with a method with zero quality in its distribution.
2. Cumulative critical regions: This is a case where slight variations on a single method basis could add up and result in a huge deviation from expected value when all the methods are combined. Consider for instance {a1,a2,a3} is a schedule to achieve task T1. Now suppose the durations of a1, a2 and a3 go slightly above their expected values but within the upper limit. So none of them is a critical region. But the duration of task T1 would cross the threshold because of additivity of durations. This is a cumulative critical region.
3. Probabilistic critical regions: When the method quality, cost and duration distributions are such that on execution, the method could fail to reach the envelope criteria with a high probability [Wagner98b,Hart90]. This means that certain portions of the distribution could have values below the lower bound or above the upper bound.

In our domain, we have considered only static critical regions i.e. the identification of critical regions is independent of the progressive results of schedule execution. Hence we do not incrementally look at the envelopes [Amant 95].

Further analysis of each of these categories of critical regions including their identification and handling as well the concept of dynamic critical regions will prove to be interesting areas for future research.

References

[Amant95] St. Amant, R.; Kuwata, Y., and Cohen, P. 1995. "Monitoring Progress with Dynamic Programming Envelopes." In Proceedings of the Seventh International IEEE Conference on Tools with Artificial Intelligence, IEEE Computer Society Press, pp. 426-433.

[Bresina94] Bresina, J.;Drummond, M; Swanson, K.,"Just-In-Case Scheduling", Proceedings of AAAI-94, Seattle, WA

[Decker95] Decker,K.; TAEMS: A framework for analysis and design of coordination mechanisms. In G. O'Hare and N. Jennings, editors, Foundations of Distributed Artificial Intelligence. Wiley Inter-Science, 1995.

[Decker93] Decker, K. and Lesser, V. Quantitative Modeling of Complex Computational Task Environments, Proceedings of the Eleventh National Conference on Artificial Intelligence, 1993.

[Hart90] Hart, D., Anderson, S. and Cohen P., 1990. "Envelopes as a Vehicle for Improving the Efficiency of Plan Execution" In Proceedings of the Workshop on Innovative Approaches to Planning, Scheduling and Control. K. Sycara(Ed.). Morgan Kaufmann. 71-76

[Wagner98b] Wagner, T.; Garvey, A.; and Lesser, V. 1998. Criteria-Directed Heuristic Task Scheduling. *International Journal of Approximate Reasoning, Special Issue on Scheduling.* To appear. Also available as UMASS CS TR-97-59.

[Wagner97c] Wagner, T.; Garvey, A.; and Lesser, V. 1997. Complex Goal Criteria and Its Application in Design-to-Criteria Scheduling. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 294–301. Also available as UMASS CS TR-1997-10.

[Wagner97e] Wagner, T.; Garvey, A.; and Lesser, V. 1997. Leveraging Uncertainty in Design-to-Criteria Scheduling. *UMASS Department of Computer Science Technical Report* TR-97-11.