Contingency Analysis in the Design-to-Criteria Scheduler

Anita Raja
Computer Science Department
University of Massachusetts

October 9, 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 which 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.

1 Introduction

The Design-to-Criteria scheduler [Wagner97a, Wagner98] is a real-time 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]. TÆMS is a compiled network of plan alternatives that describes complex problem solving processes in terms of alternate ways by which problem solving goals can be achieved. It also specifies 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 [Wagner97a] 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 [Wagner97b] 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 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, even 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 Background Work

Classical AI planning work has precluded the issue of planning for contingencies in the event of plan failure. It adopts a very narrow notion of assuming a all-or-nothing approach. Recent work in conditional planning however has focussed on solving problems which involve uncertainty by probabilistic reasoning about actions and information on the value of planning for alternative contingencies [Draper94, Kushmerick94] and using utility models [Haddaway98]. Other approaches use Partial Markov Decision Processes and decision theoretic planning approaches [Boutilier95, Dean95] which prune the search space by using domain-specific heuristic knowledge. [Onder97] describes a partial-order planner called *Mahinur* that supports conditional planning with contingency selection. They concentrated on two aspects of the problem, namely, planning methods for an iterative conditional planner and a method for computing the negative impact of possible sources of failure. We found that our work done using the Design-to-Criteria framework addresses similar questions namely

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 the execution-time performance measure and when is a specific approximation appropriate?*

[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 following ways

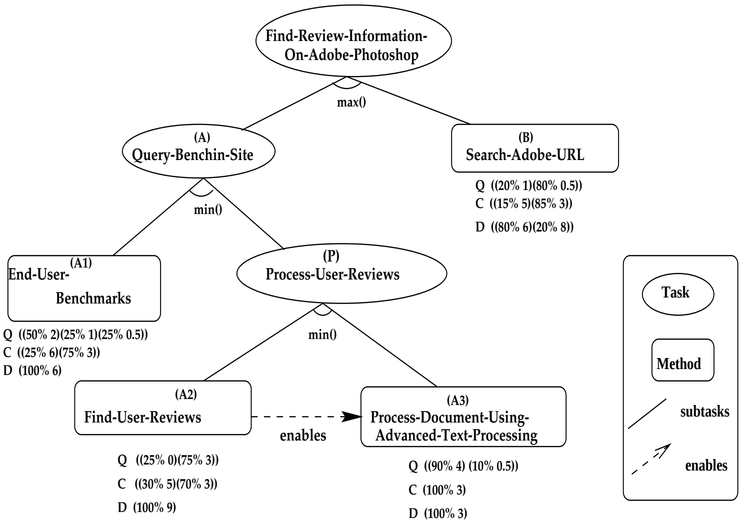

Figure 1: Gather review information on Adobe Photoshop.

1. Construction of contingent schedules in 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.
2. We constantly evaluate the user specifications with the criteria constraints to get a satisficing yet robust result.
3. 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.

3 The Expected Lower Bound and the Approximate Expected Bound

The motivation for evaluating performance measures for contingency planning is the following. “Different approximations of execution-time performance measures represent different amounts of work in handling contingency scheduling. We want to evaluate these approximations based on the amount of work and the performance trade-offs.”

3.1 An Information Gathering example

We describe a simple example which concisely captures the complexity 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), which has semantics similar to an "OR" in logic. Method End-User-Benchmarks (A1) and task Process-User-Reviews(P) [1] need to be scheduled for task Query-Benchin-Site to achieve a non-zero quality. This relationship is described by means of the min() qaf which is equivalent to an "AND" 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() qaf.

The quality, cost and duration criteria for the executable methods are described in terms of the different possible outcomes and their frequency of occurrence computed as a percentage. For instance, in Figure 1, method End-User-Benchmarks has the following quality outcome distribution Q ((50% 2)(25% 1)(25% 0.5)). It achieves quality value of 2 with probability 0.5, quality of 1 with probability 0.25 and 0.5 with probability of 0.25.

The *enables* NLE between methods Find-User-Reviews(A2) and Process-Document-Using-Advanced-Text-Processing(A3) indicates that Find-User-Reviews needs to incur a non-zero quality for Process-Document-Using-Advanced-Text-Processing to be executed. A *facilitates* NLE describes a soft relationship between methods, where a non-zero quality achieved by the facilitator method allows the expected performance of the facilitated method to improve by the degree of facilitation; however the facilitated method can still be executed with the facilitator achieving a non-zero quality or for that matter not even being scheduled.

The expected quality of results achieved by executing task Query-Benchin-Site is higher than Search-Adobe-URL and thus preferred. However it is possible that for certain products, Benchin does not contain user reviews for the product with a probability of 0.25 and hence receives a quality of 0. If Query-Benchin-Site is the only path selected, then Find-Review-Information-on-Adobe-Photoshop in turn results in zero quality. In this scenario, we would have preferred method Search-Adobe-URL which has a 100% guarantee of achieving the top-level goal even if it is of lower quality.

Lets assume the criteria requirements state that the task should achieve the maximum quality possible within 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 [Wagner97a] which compares the schedules' statistical attributes to scheduling design criteria, e.g., quality, cost, duration and uncertainty measures, provided by scheduler clients.

[1] A method which does not execute takes on a default quality value of zero.

For the sake of simplicity, we have modified and simplified the scheduler's criteria-driven evaluation mechanism to make this example and the related comparisons succinct; that is we have focussed only on the expected quality attributes of schedules and ignored the multi-dimensional and relative scaling components of the scheduler's standard utility calculation. The term *rating* in the remainder of this document will denote the expected quality of a given schedule and nothing more.

3.2 Expected Lower Bound Rating

In this paper, we will call the objective function based rating returned by the standard Design-to-Criteria scheduler the *Expected Lower Bound(ELB)* and view it as the statistical measure of the characteristics of a schedule assuming no rescheduling.

For the example described in the earlier section, we will focus on maximizing quality within a hard deadline of 18 minutes. The two possible schedules are {A1,A2,A3} and {B}. The ELB takes into account the various possible permutations of method outcomes along with the quality achieved. Figure 2 describes the computation of the ELB ratings for the schedule {A1,A2,A3}.

A1	A2	A3	Frequency	Quality
50% 2	25% 0	nil	5%*25%=12.5%	0.0
50% 2	75% 3	90% 4	33.75%	2.0
50% 2	75% 3	10% 0.5	3.75%	0.5
25% 1	25% 0	nil	6.25%	0.0
25% 1	75% 3	90% 4	16.875%	1.0
25% 1	75% 3	10% 0.5	1.875%	0.5
25% 0.5	25% 0	nil	6.25%	0.0
25% 0.5	75% 3	90% 4	16.875%	0.5
25% 0.5	75% 3	10% 0.5	1.875%	0.5

Figure 2: Each row represents a possible permutation of the quality distributions of methods A1, A2, A3 in schedule {A1,A2,A3}. The first three columns represent the possible ratings(Quality) achieved by each of the methods A1, A2, A3. The fourth column shows the probability of the particular quality distribution combination occurring and the last column shows the final quality of the schedule.

Consider the first entry of the table. It handles the case when method A1 achieves a quality of 2, which occurs with a probability of 0.5 as described in the TÆMS task structure. Method A2 achieves a quality of 0 with probability 0.25. [2] The probability of the methods achieving these qualities simultaneously in a single execution is 0.125, given in column 4. The expected quality of the schedule {A1,A2,A3} is 0 in this case, described in column 5. The duration and cost distributions and their expected values are computed in a similar fashion. The ELB ratings for schedules {A1,A2,A3} and {B} are given below.

[2]Failure of A2 (Quality = 0) automatically results in zero quality for the schedule due to the specifics of the concerned task structure. Hence the quality of A3 is a not a determining factor and is represented by nil.

1. {A1,A2,A3}: Rating: 0.97 (Expected Quality)
   Quality : (25% 0.0) (24% 0.5) (17% 1.0) (34% 2.0)
   Duration : (100% 18)
2. {B}: Rating 0.6 (Expected Quality)
   Quality : (20% 1) (80% 0.5)
   Duration: (80% 6) (20% 8)

This example is amenable to the contingency-tree style calculation as shown in Figure 3 , but the general case is not. In the case of task structures with significant levels of complexity, the computation of the ELB by the Design-to-Criteria scheduler is not based on a contingency-tree style analysis, due to the combinatorics of the general scheduling problem. In this particular example, the ELB would then be an underestimate of expected schedule quality and the ratings would be as follows.

1. {A1,A2,A3}: Rating: 0.72 (Expected Quality)
   Quality : (44% 0.0) (18% 0.5) (13% 1.0) (25% 1.0)
   Duration : (100% 18)
2. {B}: Rating 0.6 (Expected Quality)
   Quality : (20% 1) (80% 0.5)
   Duration: (80% 6) (20% 8)

We use limited tree expansion in situations such as these where it is not precluded by the combinatorics of the actual scheduling instance. In other instances, where the complexity of the task structure is significant, heuristics are used for approximating schedule performance to be able to handle the combinatorics.

The schedule {A1,A2,A3} is chosen and executed since it has the best expected lower bound rating of 0.97. 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 3) 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.

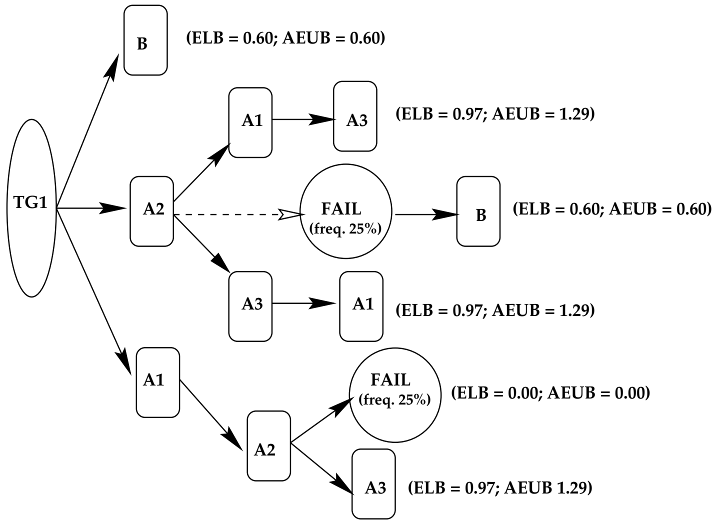

Figure 3: Schedule options for example one with (schedule rating) values.

3.3 Approximate Expected Upper Bound, Approximate Expected Bound, Critical Task Execution Regions

In our example, task A2 has an enables non-local effect [Wagner97a] as well as a 25% chance of failure within its distribution. We hence predict that task A2 could potentially be a *critical task execution region(CTER)*. A *CTER* 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 *CTER*, 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 i.e for method A2, Q (100% 3). The Design-to-Criteria scheduler is reinvoked with the modified task structure and reschedule. The following are the ratings returned by the scheduler.

1. $\{A1, A2^{success}, A3\}$: Rating 1.29 (Expected Quality)
   Quality : (32% 0.5)(22% 1.0)(45% 2.0)
   Duration: (100% 18)
2. {B} : Rating 0.6 (Expected Quality)
   Quality: (20% 1) (80% 0.5)
   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 $\frac{1.29-0.97}{0.97} * 100 = 33$ % improvement in quality with respect to the expected lower bound rating is significant. This 33% 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 *CTER*.

The tree structure in Figure 3 presents all the options of schedule generation that will meet the criteria of a duration limit of 18 minutes. 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.

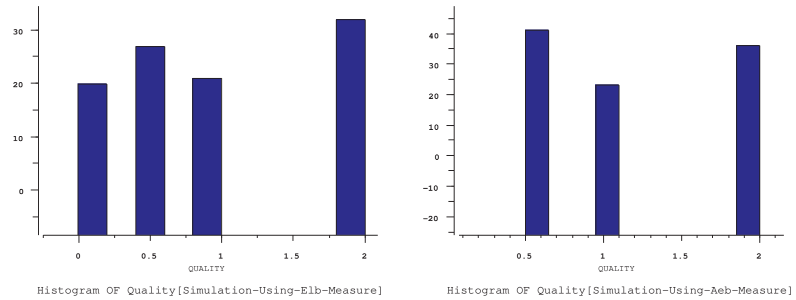

Figure 4: Statistical Performance of schedules with highest ELB and AEB

The bar graph on the left of Figure 4 shows the statistical performance of the schedule with the highest ELB for 100 simulation runs. A simulation run is a simulated execution of the schedule with highest ELB and the actual quality, cost and durations values are averaged over the number of simulations to obtain a statistical rating. We note that the schedule fails to achieve any quality about 20% of the time. The mean quality achieved by using this performance measure is 0.98.

The bar graph on the right Figure 4 describes the statistical performance data for the schedule with the highest AEB over 100 simulation runs. Here 100 simulated executions of the schedule with the highest AEB rating produce the data. As seen in the histogram, the quality of the schedule with the best AEB rating is always a non-zero value due to the built-in contingency and the mean quality achieved here is 1.96.

4 Performance Measures

In this section we try to formalize a general theory relating to the concepts on contingency discussed in the previous section. The question we strive to answer formally here is the following: *What performance measure is the most appropriate estimator of the actual execution behavior of a schedule the criteria conditions?*

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 sequence 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 *CTER* 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 [Wagner97a]. It is a expected rating because it is computed on a statistical basis taking quality, cost and duration distributions into account.

Given a schedule $s = (m_{1..} m_n)$, we can compute the $ELB\{s\}$, for a criteria namely quality in the following manner:

Let the possible values for quality of the schedule s be $q1..qp$. For each schedule quality qk, we use the corresponding set of outcomes $m_{1j_{qk}}, m_{2j_{qk}}, ...m_{nj_{qk}}$ and their frequencies in s and $\sum_{q_k} prod_{i=1}^{n} f_{ij_{qk}})$ will give the expected frequency of q_k. A similar computation is done for the cost and duration criteria values of schedule s. As mentioned earlier, the ELB computation in this paper is an underestimate of the expected value as the computation has been restricted by the combinatorics of the general scheduling problem.

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 $\overline{s_{ij}^{cr}} = (m_1, m_2..\overline{m_{ij}^{cr}}..m_n)$.

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

For our Information Gathering example, we see that $\frac{ELB(\{\overline{A2},A1,A3\})-ELB(\{A1,A2,A3\})}{ELB(\{A2,A1,A3\})} \geq$ 0.3. Hence there is at least an 30% 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 *CTER's*, we call it the Approximate Expected Upper Bound. Generalizing this formula for k *CTER's* $m_{i_1j_1}...m_{i_kj_k}$,

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

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 it is unrealistic to use it for 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, $ELB((m_1...m_n)) \gg ELB((m_1^e, m_2^e...m_i^e, m_{i+1}..m_n))$, then a new schedule is constructed based on the partially complete schedule $\{m_1^e, m_2^e, ...m_i^e\}$. The determination of whether the ELB rating of a particular schedule is significantly greater than the ELB rating of another is done using

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 a *CTER* and using expected lower bound of the new stable schedule for methods following the *CTER*. This is limited contingency analysis at *CTER's*.

Consider a schedule s of n methods $m_1, m_2..m_i..m_n$. Now suppose m_{ij} is a *CTER* 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 described when the AEB was introduced in a previous section.

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 *CTER's* named $m_{i_1j_1}^{cr}, m_{i_2j_2}^{cr}...m_{i_kj_k}^{cr}$. Let the recovery path available at each *CTER* m_{ij}^{cr} be s_{ij}^{r} and each m_{ij}^{cr} occurs with frequency f_i^{cr}. The AEB of the entire schedule is described recursively as $AEB = ELB(m_1...m_{ij}^{cr}, l_1, ...l_k) * f_i^{cr} + AEB(m_1...\overline{m_{ij}^{cr}}, m_{i+1}, ...m_n) * (1 - f_i^{cr})$ which can be expanded out as follows:

$$
\begin{aligned}
AEB = {} & f_1^{cr} * ELB(m_1...m_{i_1-1}, m_{i_1j_1}^{cr}, l_{a1}...l_{b1}) \\
& + (1 - f_1^{cr}) * f_2^{cr} * ELB(m_1...\overline{m_{i_1j_1}^{cr}}...m_{i_2j_2}^{cr}, l_{a2}...l_{b2}) \\
& + ...(1 - f_1^{cr}) * ... * (1 - f_{k-1}^{cr}) * f_k^{cr} * ELB(m_1...\overline{m_{i_1j_1}^{cr}}...\overline{m_{i_2j_2}^{cr}}...\overline{m_{i_3j_3}^{cr}}...m_{i_kj_k}^{cr}, l_{ak}...l_{bk}) + \\
& (1 - f_1^{cr}) * (1 - f_2^{cr}) * ... * (1 - f_k^{cr}) * \underbrace{ELB(m_1...\overline{m_{i_1j_1}^{cr}}...\overline{m_{i_2j_2}^{cr}}...\overline{m_{i_kj_k}^{cr}}...m_n)}_{AEUB}
\end{aligned}
$$

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 *CTER's*, 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 [Wagner98].

5 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 :

1. 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.
2. Suppose the scheduler evaluates k schedules to decide which is the best schedule, where $s_k = (m_1^k ... m_n^k)$ and let S be the set of all k schedules.
3. s^b has the highest ELB in S.
4. Let $S_{rem} = S - s^b$. Then $ELB(s^b) \geq ELB(s)$ for all $s \in S_{rem}$.
5. Let S_{rem}^b be the set of $s \in S_{rem}$ such that $AEUB(s) \gg ELB(s^b)$. If $S_{rem}^b \neq \phi$, then we compute the $AEB(s)$ for each $s \in S_{rem}^b \bigcup s^b$.
6. The new best schedule s_{aeb}^b is the one with with the highest AEB. s_{aeb}^b is guaranteed be more robust.

Identifying *CTER'S*

The AEB is a better estimate than the ELB when there is uncertainty in the schedule, i.e., there are *CTER's* in the schedule and there is a possibility for contingency plans. Earlier we defined *CTER's* as those regions in the schedule which could potentially lead to degradation in the expected performance. 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 A2 has the following quality distribution Quality :(25% 0)(75% 3) which means there is a 25% chance of failure. This makes it a candidate *CTER*.
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 not only has a failure possibility in its distribution, it also enables method A3. Thus, it becomes imperative to evaluate A2 as a *CTER*.
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 scheduled(using the ELB measure) redundant activities to overcome *CTER's*, 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 *CTER*, 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. This issue of

redundancy is discussed in detail later on in the paper. 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,B} as there is enough time to reschedule B in case of failure of A2. The AEB computation also chose the schedule {A2,A1,A3,B} 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 *CTER's*. 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 *CTER*. 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 *CTER*. 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.

Redundancy in the Design-to-Criteria scheduler:

An interesting extension of the evaluation in our example is to look at schedules that are produced to resolve uncertainty which in some cases instead of assuming success, assumes failure.

Suppose in the Information Gathering example the results of task B is a subset of the results of task A, if task A is executed successfully. In other words the search at the Adobe site will provide only redundant information, if the Benchin site has been successfully queried. Let us assume that the new criteria is to maximize quality, a soft duration deadline of 18 minutes and a hard duration deadline of 25 minutes.

The Design-to-Criteria scheduler would then present the schedule {A2,A1,A3,B} as it would have the highest ELB. So if A2 fails, execution of B would ensure that the high level goal is achieved. But the ELB computation doesn't assume rescheduling if A2 succeeds which eliminates the need to execute method B. We know $ELB(\{A2, A1, A3, B\})$ would never be better than $ELB(\{A2, A1, A3\})$ if A2 succeeded because method B is redundant and its only effect is to increase the duration of the schedule which decreases the ELB rating. In general, if the ELB criteria attaches any significance to the duration of the schedule, then the removal of actions from the schedule due to the results of prior actions making this action redundant will always increase the ELB rating.

The AEB calculation for schedules that have built-in contingencies, both successful and failure action evaluation has to be modified. Normally, contingency analysis is done for the failure region. In this case where the contingency schedule for failure is a subset of the existing schedule, one needs to do contingency analysis for both success and failure possibilities. We extend the formula described in the definition of AEB. Let $m_1, m_2...m_{ij}^{cr}, k_1, ..k_p, l_1, ..l_q, m_{i+1}..m_n$ be a schedule s of n methods with a critical region m_{ij}^{cr} which occurs with frequency of failure f_i^{cr}. Let the recovery path available at critical region m_{ij}^{cr} be $l_1, l_2...l_q$ and suppose its a subset of $k_1, k_2..k_p$ where $k_1, k_2..k_p$ produces quality only if m_{ij}^{cr} succeeds and the quality produced by $l_1, l_2...l_q$ is independent of the success of m_{ij}^{cr}. The AEB of the entire schedule is described recursively as $AEB(s) = (1 - f_1^{cr}) * AEB(m_1, m_2...\overline{m_{ij}^{cr}}, k_1, ..k_p, m_{i+1}, ..m_n) + (f_1^{cr}) * ELB(m_1, m_2..m_{ij}^{cr}, l_1..l_q, m_{i+1}...m_n)$

So in schedule $A2, A1, A3, B$, the exact evaluation of the schedule would be one which takes both $A2^{success}$ and $A2^{failure}$ into consideration. If A2 is successful, then the methods related to failure of A2 should be eliminated (method B in this case) while rating $A2^{success}$. Likewise, if A2 fails, methods associated with the success of A2 namely A1,A3 should be eliminated while rating $A2^{failure}$. So $AEB(A2, A1, A3, B) = ELB(A2^{success}A1, A3) + ELB(A2^{failure}B)$.

6 Experimental Results

Using the measures described above, effective contingency planning is a complex process. It involves taking into account a number of factors namely task relationships, deadlines, availability of alternatives, user-directed quality, cost and duration criteria.

As a part of the evaluation process, we will try to describe the characteristics of the objective function as well as the characteristics of the task structures for which it would be advantageous to do contingency planning. We will also explain why these characteristics affect the performance.

We performed the evaluation by randomly generating task structures with varied task structure characteristics and doing contingency analysis by varying the multi-attributed objective function. We used our prior knowledge of the potential of the performance measure to seed the search for the types of task structures which would benefit from contingency analysis. Since method failure is a crucial factor for the contingency analysis argument, we have focused our attention on two factors namely, the effects of failure location and failure intensity(probability of failure). We used 10 randomly generated task structures with specific characteristics and specific objective functions. Figure 5 shows three such randomly generated task structures used in the evaluation.

The results from the performance evaluation are shown in Figure 6. For each task structure. 30 simulated executions were performed using the schedule with the highest ELB and similarly for the schedule with highest AEB. The schedule qualities achieved

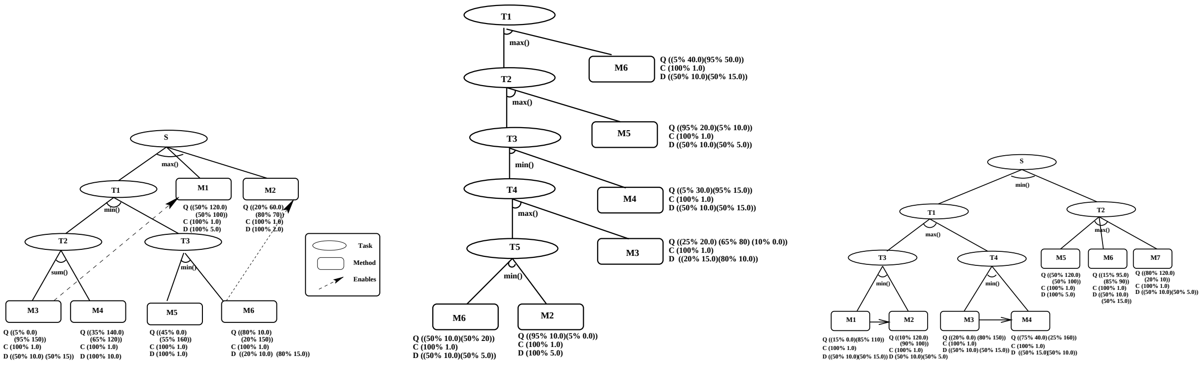

Figure 5:

were then normalized by the highest execution quality achieved in the 60 executions and the average quality from non-contingency(ELB-based) schedule executions and quality from contingency(AEB-based) schedule executions were recorded. The above steps were repeated for each of the 10 task structures and then we finally averaged the non-contingency execution quality averages and contingency execution quality averages represented by Ave. Quality Non.Cont.Schldg and Ave.Quality Cont.Schldg respectively.

Failure		Ave. Quality	Ave. Quality	Performance
Location	Intensity	Non.Cont. Schdlg	Cont. Schdlg	Improvement
Early	Medium	0.70971	0.80198	11.5%
Medium	Medium	0.76280	0.82939	8.02%
Late	Medium	0.61334	0.72694	15.63%
Medium	Low	0.80845	0.84331	4.13%
Medium	Medium	0.66559	0.75785	12.17%
Medium	High	0.59802	0.65194	8.27%

Figure 6: The first 2 columns show the variables being tweaked namely Failure Location and Failure Intensity. The third and fourth column show the normalized execution quality without contingency planning and with contingency planning averaged over 300 simulated executions. The last column shows the performance improvement achieved by contingency analysis.

Failure Location refers to the position of critical method(s) in a task structure and hence in the schedule. We have classified Failure locations into 3 namely, Early, Medium and Late failure. Similarly, Failure Intensity refers to the probability of a method failing and we have classified it to be Low, Medium and High where 1-10is Medium and 41

The first three rows show the performance of schedules with varying the Failure Location factor. The Failure Intensity was kept constant at Medium and the objective function was focussed on maximizing quality while trying to keep schedule duration to a minimum. No hard duration deadlines or quality thresholds were specified.
Firstly, we note that contingency planning gives a significant improvement in performance in all three cases. We see that when failure is early in the schedule, both non-contingency

and contingency perform relatively well. This is because the non-contingency case can insert redundant methods in the schedules to handle failures. The contingency case performs better because it plans ahead for the failures and no time or quality is lost in performing redundant methods. When failure is in the middle regions, performance is scaled up. The non-contingency case applies its redundancy heuristic to cover for method failures and the contingency case executes schedules which have good average performance with and without failures. When failure is late in the schedule, performance degrades in both cases. The non-availability of viable alternatives late in the schedule execution is the reason for this. The non-contingency case was often found to take a lower quality yet safer(minimal failure) route in most of the runs. Contingency planning however tries to find contingent plans which would be a safe path but this was not always a possibility. This explains the drop in performance from the Medium Location case but the significantly better performance compared to non-contingency scheduling in similar settings.

The last three rows show the performance of schedules with varying the Failure Intensity factor. The Failure Location was kept constant at Medium level and again the objective function were focussed on maximizing quality while trying to keep schedule duration to a minimum. No hard duration deadlines or quality thresholds were specified.
Once again, we note that contingency planning gives a significant improvement in performance in all three cases. We see that when Failure Intensity is low, both non-contingency and contingency perform very well.In fact this is the best case for all the 6 cases. Also there is not a big difference in performance between the two cases, there is only a 4% increase in performance. This is because failure occurs at a vary low rate and hence contingency planning is really not necessary. When Failure Intensity is medium, contingency regains its advantageous position by being able to handle both the success and failure of critical methods. When Failure Intensity is very high, performance is at its worst relative to the other cases and this is because all paths are riddled with highly critical points nad the probability of the plan completely failing for all possible plans is pretty high. Even in this case we see that contingency performs better than the non-contingency case because if there is a path, namely by rearranging methods and trading off nle's, which can achieve quality, contingency planning would find it while the non-contingency analysis will not.

We now describe the characteristics of task structures which make it advantageous to perform contingency planning.

1. Methods in task structures should have possibility of failure in their distribution.
2. There could be multiple methods which could fail in a single task structure.
3. Task structures should contain alternate paths with significant difference in performance. For instance when one path has high quality and also high risk of failure and another path is low quality but has no failure, it would be useful to do contingency analysis.
4. A possibility of moving failure methods forward(absence of associated hard nle's) would further the potential of contingency analysis.
5. Presence of an alternate path with low quality, low cost, low duration and low uncertainty.

6. Dependence of methods with good average performance on critical methods.(enables nle from a critical method to a non-critical method.)

The following characteristics of objective function which augment contingency planning.

1. The objective function could specify a hard deadline, and emphasis should be given to either the quality or duration slider.
2. The deadline should also provide enough time for contingency analysis.
3. Giving relatively equal importance to the quality goodness and duration sliders and maxing the meta goodness slider.
4. Setting relatively equal importance to the meta goodness and meta duration sliders if a deadline is specified.

7 Conclusions and Future Work

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. We also described the characteristics of the task structures and objective functions for which contingency analysis is advantageous.

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 our domain, we have considered only static critical task execution regions i.e. the identification of critical task execution 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 task execution regions including their identification and handling as well the concept of dynamic critical task execution regions will prove to be interesting areas for future research.

We plan to build a front end to this system which will classify task structures and their objective functions to viable and non-viable structures for contingency analysis. This would improve the cost the performance ratio of scheduling and executing plans.

We also plan to determine the relationship between number of reschedulings occuring in a single execution and how well the schedules with best ELB and best AEB ratings approximate the actual execution performance. In other words we would like to compare the plan that is actually executed to the plan that was suggested by the contingency planning.

We also plan to compare the performance of tree based ELB computation versus the underestimate ELB computation which uses approximation heuristics.We would like to infer the degree of task structure complexity which makes it uses to perform tree based ELB computation which is very accurate versus using the non-tree case which is less accurate but suitable for handling the combinatorics of a typical real world scheduling problem.

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.

[Boutilier95] Boutilier, C.;Dean, T.; and Hanks, S. *Planning Under Uncertainty: Structural Assumptions and Computational Leverage* In Proceedingsof Proc. 3rd European Workshop on Planning (EWSP'95).

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

[Dean95] Dean, T.;Kaebling, L; Kirman, J.; Nicolson, A, *"Planning under time constraints in stochastic domains"*, Proceedings of AI-95.

[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.*

[Draper94] Draper, D.; Hanks, S.; and Weld, D., *"Probabilistic Planning with Information Gathering and Contingent Execution"* In Proceedings of the Second International Conference on Artificial Intelligence Planning Systems (AIPS-94) pages 31-36.

[Haddaway98] Haddaway, P.; Hanks, S. 1998. *"Utility models for goal-directed decision-theoretic planners"* Computer Intelligence, Volume 14, Number 3, 1998.

[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.

[Kushmerick94] Kushmerick, N.; Hanks, S.; and Weld, D. "An Algorithm for Probabilistic Planning" 1994. In *Proceedings of Artificial Intelligence,1994* (Short Version AAAI-94).

[Onder97] Onder, N.; Pollack, M. "Contingency Selection in Plan Generation" 1997. In *Proceedings of the Fourth European Conference on Planning* (ECP'97)

[Peot92] Peot, M.; Smith, D. "Conditional nonlinear planning" 1997. In *Proceedings of First International Conference on AI Planning Systems* pages 189-197, 1992.

[Wagner98] 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.

[Wagner97a] 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.

[Wagner97b] 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.

[Williamson94] Williamson, M., and Hanks, S., 1994 "Optimal Planning with a Goal-Directed Utility Model" In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems* (AIPS-94) pages 176-181.