Toward Robust Agent Control in Open Environments [*]

Anita Raja, Victor Lesser and Thomas Wagner
Computer Science Department
University of Massachusetts

UMass Computer Science Technical Report 1999-059

March 6, 2000

Abstract

Open environments are characterized by their uncertainty and non-determinism. This poses an inevitable challenge to construction of agents which need to operate in such environments. The agents need to adapt their processing to available resources, deadlines, the goal criteria specified by the clients as well their current problem solving context in order to survive. Our research focuses on constructing a framework for robust agent control, using a soft-real time scheduling approach which satisfices all aspects of problem solving. In this paper, we evaluate the performance of our heuristic-based approach using the performance of the policy generated by an optimal controller as the benchmark.

Keywords: Selection and Planning, Real-time performance, Designing Agent Systems

[*] Effort sponsored by 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 number F30602-99-2-0525 and by the National Science Foundation under Grant number IIS-9812755 and number IRI-9634938. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, NSF, USAF, or the U.S. Government.

1 Introduction

It is paramount for agent-based systems to adapt to the dynamics of open environments. The agents need to adapt their processing to available resources, deadlines, the goal criteria specified by the clients as well their current problem solving context in order to survive in these environments. In this paper, we describe a framework for equipping agents to perform efficiently by ensuring robust control. We evaluate the performance characteristics of this heuristic based controller by comparing it to that of an optimal policy generated by an MDP-based meta-controller.

We reason about and model an agent's problem solving activities using TÆMS (Task Analysis, Environment Modeling, and Simulation), a domain independent task modeling framework. TÆMS models are used in multi-agent coordination research [5] as well as Cooperative-Information-Gathering [13, 11], collaborative distributed design [5], and intelligent environments [10]. Typically an agent represents domain problem solving actions in TÆMS, possibly at some level of abstraction, and then passes the TÆMS models on to agent control problem solvers like the multi-agent coordination modules or the Design-to-Criteria (DTC) scheduler. We present the task modeling semantics in Section 2.

DTC scheduling [16] is the soft real-time process of finding an execution path through a hierarchical task network such that the resultant schedule meets certain design criteria, such as real-time deadlines, cost limits, and quality preferences. It is the heart of control in agent-based systems such as the resource-Bounded Information Gathering agent BIG [11] and the multi-agent Intelligent Home [10] agent environment.

The general DTC scheduling process is designed to cope with exponential combinatorics and to produce results in soft real-time. However, DTC's somewhat myopic approximation and localization methodologies do not consider the existence of recovery options or their value to the client. In the general case, explicit contingency analysis is not required. In the event of a failure, the scheduler is reinvoked and it plans a new course of action based on the current context [18, 7]. In hard deadline situations, such as those in mission critical systems [12] however, the scheduler may not be able to recover and employ an alternative solution path because valuable time has been spent traversing a failed solution path. We have recently expanded the DTC heuristic scheduler to make more informed decisions about the choice of schedules using contingency analysis. In particular, the chosen schedule takes into account the fact that rescheduling can occur. Our uncertainty based contingency analysis tools pre-evaluate the likelihood of recovery from a particular path and factor that into the utility associated with a particular schedule. The improved estimates can result in the selection of a different schedule, possibly one that leads to higher quality results with greater frequency resulting in improved real-time performance. We return to contingency analysis in Section 3.

Obviously, in the best of all possible worlds, instead of generating a schedule where we have to reschedule at failure points, we would like to construct an optimal meta-control policy which prescribes the next best action to take based on performance characteristics of the most recently executed primitive action. For most reasonable size task structures the computational overhead of constructing this policy online is unrealistic. However, we would like to see how well the contingency analysis approach performs relative to an optimal policy. In Section 4, we describe an approach for constructing such an optimal policy from our TÆMS task graph representation.

Experimental results illustrating the strength of contingency analysis, relative to Design-to-Criteria's myopic view are discussed in [18]. In Section 6, we discuss the performance of contingency analysis relative to an optimal controller's global view, for certain classes of task structures.

2 The TÆMS Modeling Language

An agent models its domain problem solving actions in
TÆMS by framing its activities in the short to medium term view required to construct the task structure. TÆMS task structures capture several important features that facilitate the agent's problem solving. These include the top-level goals/
objectives/ abstract-tasks that the agent intends to achieve as well as one or more of the possible ways that they could be achieved expressed as an abstraction hierarchy whose leaves are basic action instantiations, called methods. It models a precise, quantitative definition of the degree of achievement in terms of measurable characteristics such as solution quality and time. Information about task relationships that indicate how basic actions or abstract task achievement effect task characteristics (e.g., quality and, time) elsewhere in the task structure, the resource consumption characteristics of tasks and how a lack of resources affects them are also embedded in the task structure.

Consider the TÆMS task structure shown in Figure 1. It is a con-

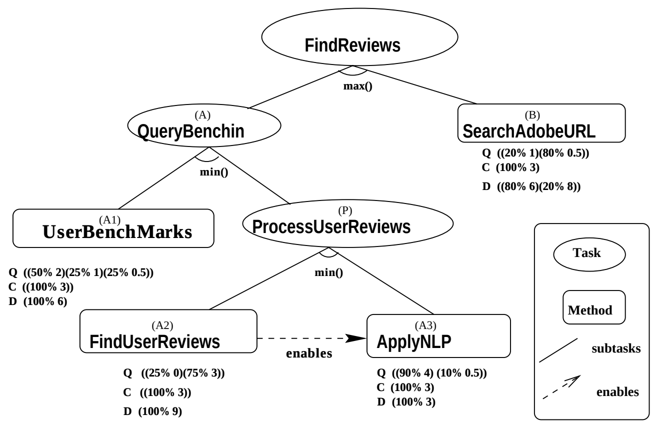

Figure 1: Information Gathering Task Structure

ceptual, simplified sub-graph of a task structure emitted by the BIG [11] information gathering agent; it describes a portion of the information gathering process. The top-level task is to obtain reviews on Adobe Photoshop. The top-level task is decomposed into a subtask *Query-Benchin (A)* and a method *Search-Adobe-URL (B)*. Methods are primitive actions which can be scheduled and executed and are characterized by their expected quality, cost and duration distributions. For instance, the quality distribution of method *User-Benchmarks* indicates that 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. Quality is a deliberately abstract domain-dependent concept that describes the contribution of a particular action to overall problem solving. Thus, different applications have different notions of what corresponds to quality. Duration describes the amount of time that the action modeled by the method will take to execute and cost describes the financial or opportunity cost inherent in performing the action. With the recent addition of uncertainty modeling, the statistical characteristics of the three dimensions are described via discrete probability distributions associated with each method. *Find-Reviews* can be achieved by either completing task *Query-Benchin* successfully or executing the method *Search-Adobe-URL* or both and the maximum of the qualities is propagated to *Find-Reviews*. This is indicated by the max() quality accumulation function(qaf), which defines how performing the subtasks relate to performing the parent task. The *min()* qaf under *Query-Benchin* indicates that either executing method *User-BenchMarks (A1)* or completing task *Process-User-Reviews (P)* or both may be employed and the minimum of the qualities (indicated by the min() qaf) will be propagated to *Query-Benchin*. The *enables* arc between *Find-User-Reviews (A2)* and *Apply-NLP (A3)* is a non-local-effect (nle) or task interaction; it models the fact that user reviews need to be obtained in order to perform sophisticated extraction techniques on the documents.

Two different schedules for achieving the top-level goal of the task structure, produced for two different sets of design criteria, are shown in Figure 2. Schedule A is constructed for a client who needs a high quality solution, requires the solution in twenty minutes or less, and who is willing to pay for it. Schedule B is constructed for a client looking for a cheap and quick solution. This example illustrates the notion of quantified choice in TÆMS and how the DTC methodology leverages the quantification to build different schedules for different contexts.

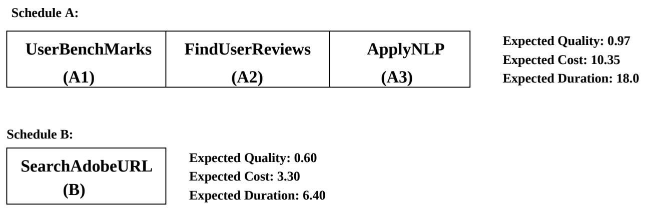

Figure 2: Different Schedules Produced for Different Design Criteria

3 Uncertainty-based Contingency Analysis

Design-to-Criteria is the process of coping with exponential combinatorics to produce schedules in soft real-time that meet a particular set of design criteria and hard constraints like deadlines or cost limitations. Because the scheduling problem entails enumerating the alternative different ways to achieve the top level task, and determining a sequencing for each different way for task achievement, the combinatorics are pronounced ($\omega(2^n)$ and $o(n^n)$, where n is the number of primitive actions being scheduled) and finding an optimal solution is not generally possible even for a small task structure. The scheduler controls the combinatorics through a satisficing methodology described in [16].

In order to address resource limitations and to produce schedules in interactive time, DTC exhibits a somewhat myopic view in its schedule construction. It does not consider the existence of recovery options or their value to the client. In the general case, explicit contingency analysis is not required since the scheduler is reinvoked in the event of failure and a new course of action is taken based on the current context. However, in situations where hard deadlines exist, a mid-schedule failure may preclude recovery via rescheduling because sufficient time does not remain to explore a different solution path. In these situations, a stronger analysis that considers the existence of possible recovery options may lead to a better choice of schedules. To address such situations, we have developed a contingency analysis methodology [15, 18] that functions as an optional back-end on the Design-to-Criteria scheduler.

The contingency analysis algorithms operate by examining the highly-rated candidate schedules produced by the scheduler and exploring failure / recovery scenarios for each schedule in the set. The contingency analysis tools also perform more detailed reasoning about the placement of methods within a schedule in light of the existence of recovery options. For example, recovery for a given schedule may be possible *iff* some critical method m is performed first rather than second. The standard scheduler is weakly biased toward moving uncertain methods earlier in the schedule, but the determination is *local*, based only on the attributes of the method in question, whereas the method movement explored in the contingency analysis also takes into account the benefits of method movement from a recovery perspective.

This work in contingency analysis of schedules is closely related to recent work in conditional planning. However, the planning-centric research focuses on solving problems which involve uncertainty by probabilistic reasoning about actions and information on the value of planning for alternative contingencies [6, 9] and using utility models [8]. Other approaches use partial Markov decision processes and decision theoretic planning approaches [2, 3] which prune the search space using domain-specific heuristic knowledge. [14] describes a partial-order planner called *Mahinur* that supports conditional planning with contingency selection.

To better illustrate the power of contingency analysis, consider the simple example in Figure 1. Lets assume the client design criteria specifies that the task should achieve the maximum possible quality within a hard deadline of 18 minutes. The DTC scheduler first enumerates a subset of the alternatives that could achieve the high level task. A subset of these alternatives are selected and schedules are created using the one-pass method-ordering techniques identified in [18]. The set of candidate schedules are then ranked using the multi-dimensional evaluation mechanism [16] which compares the schedules' statistical attributes to the client design criteria.

The two possible schedules which adhere to the client specified criteria are shown in Figure 2. The expected lower bound(ELB) [15] is the expected rating of a given schedule assuming no rescheduling. In this example, since the criteria is simply to maximize quality, the ELB is equal to the expected quality[1]. {A1,A2,A3} has an ELB of 0.97 while schedule {B} has an ELB of 0.6. Since {A1,A2,A3} has the highest ELB it is chosen and executed. Suppose A1 executes successfully, but A2 fails (i.e. it results in 0 quality), which it does 25% of the time. Then A3 cannot be executed because it is not enabled (A2 failed) but there is no time left to reschedule and attempt method B because there is insufficient time left to execute method B before the deadline.

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 ELB ratings. Hence the contention is that there is no difference in performance between the two. However with more detailed evaluation of the schedules, it is clear that {A2,A1,A3} allows for recovery and contingency scheduling which schedule {A1,A2,A3} does not permit 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 *Find-Reviews*. 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.

A *critical task execution region*(CTER) is a method that has the potential to seriously degrade the performance characteristics of the overall schedule if it should fail. The *approximate expected upper bound* (AEUB) is the expected rating(quality in this case) of schedules, computed with the *CTER*'s criticality removed. For this example, A2 is a potential CTER because it is an enabler of method A and it has a relatively high failure rate(25%). We will now verify if A2 is a CTER. First, we remove the failure possibility from the performance characterization of A2 and replace method A2's 25% chance of quality 0 with the expected value of the distribution.

[1]The example exhibiting these characteristics was deliberately chosen to simplify the discussion. The contingency analysis tool is capable of handling the general client criteria specification.

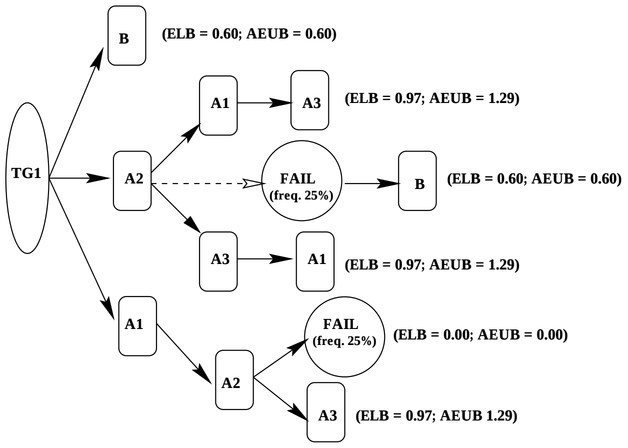
TG1
B
(ELB = 0.60; AEUB = 0.60)
A1
A3
(ELB = 0.97; AEUB = 1.29)
A2
FAIL
(freq. 25%)
B
(ELB = 0.60; AEUB = 0.60)
A3
A1
(ELB = 0.97; AEUB = 1.29)
A1
FAIL
(freq. 25%)
(ELB = 0.00; AEUB = 0.00)
A2
A3
(ELB = 0.97; AEUB 1.29)

Figure 3: Schedule Options for IG Task (Figure 1) where Ratings are Expected Qualities

Method A2 hence is assigned a quality of 3, with a probability of 1, i.e. for method A2, Q (100% 3). Then the DTC scheduler is reinvoked with the modified task structure and rescheduled. The following are the AEUBs returned by the scheduler.

1. $\{A1, A2^{success}, A3\}$: AEUB 1.29
   Quality : (32% 0.5)(23% 1.0)(45% 2.0)
   Duration: (100% 18)
2. {B} : AEUB 0.6
   Quality: (20% 1) (80% 0.5)
   Duration: (80% 6) (20% 8)

The AEUB statistic describes performance expectations if failure is not possible. The relationship between the AEUB and the ELB is a clue to the importance of the potential *CTER* to the overall schedule. In this case, the schedule {A1,A2,A3} now has an expected quality value of 1.29. The $\frac{1.29-0.97}{0.97} * 100 = 33$ % improvement in quality with respect to the ELB is significant. This 33% improvement in quality confirms that the possibility of failure in method A2 significantly decreases the rating of schedule {A1,A2,A3} and A2 is indeed a CTER. The next step is to consider the optional schedules for the original task structure to neutralize the effect of this *CTER*.

The approximate expected bound(AEB) is the schedule rating with rescheduling only at CTERs and using the ELB of the new stable schedule following the CTER. The tree structure in Figure 3 presents all possible scheduling options, including recovery scenarios, that meet the hard deadline 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 fails. Thus 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 $= 1 * 0.60 = 0.60$. Note that with this detailed analysis it is clear that schedule $\{A2, A1, A3\}$ has better expected performance than $\{A1, A2, A3\}$. However, the ELB computation of the Design-to-Criteria scheduler returns an identical ELB for both $\{A1, A2, A3\}$ and $\{A2, A1, A3\}$ as it does not take into account the recovery options present within $\{A2, A1, A3\}$.

The contingency analysis algorithms use the DTC scheduler to explore mainly the "good" portions of the schedule solution space – that is those schedules that best address the client's design criteria. This serves to constrain the computation and reduces the combinatorics from their general upper bounds. More importantly, the algorithm presented here is amenable to future research in bounding the algorithm directly, which would enable the contingency analysis approach to operate in interactive time, as does the underlying DTC scheduler.

4 Markov Decision Processes as optimal meta-controllers

Effective contingency planning is a complex process. It involves taking into account a number of factors, including task relationships, deadlines, the availability of alternatives, and client design criteria (i.e., quality, cost, duration, and certainty trade-offs). The use of DTC as an oracle enables the contingency analysis tools to cope with the combinatorics of the general scheduling problem.

In a perfect world scenario, an optimal meta-control policy would determine the best possible method to execute at each instant so as to achieve the desired high-level goal while optimizing the criteria and resource specifications. The drawback with such an approach is that for most reasonable size task structures the computational overhead of constructing this policy online is infeasible. However, we would like to see how well the contingency analysis approach performs in relation to optimal, assuming the policy is computed off-line.

Our approach for constructing an optimal policy is to define the problem as a finite-horizon Markov Decision Process(MDP) which tries to maximize its expected accumulated reward i.e. The MDP provides an optimal policy for achieving the high level goal given the criteria (quality, cost, duration) specification. It is a finite-horizon MDP because a primitive action can be executed only once in a particular execution path and hence there are no loops. MDPs are widely used in artificial intelligence as a framework for decision-theoretic planning [3, 4], reinforcement learning [1] and reactive control techniques [19].

As mentioned earlier, the Design-to-Criteria scheduling problem is framed in terms of a TÆMS task network, which imposes structure on the primitive actions and models the task interactions. The execution characteristics of primitive actions are modeled in terms of quality, cost and duration distributions. The following are some of the functional differences between the TÆMS framework and the MDP framework.

1. TÆMS does not represent the actual effects of the individual alternative paths. In other words, it does not carry through the implications of choices. The MDP framework, on the other hand, explicitly describes the primitive actions and their precise execution characteristics.

2. TÆMS specifies constraints on an ordering rather than explicitly representing the implications of the ordering. Consider, for instance, the primitive methods *User-Bench-Marks*, *Find-User-Reviews*, *Apply-NLP* and
   Search-Adobe-URL in Figure 1. The only constraint on ordering specified by the TÆMS task structure is that execution of method *Find-User-Reviews* should precede that of *Apply-NLP*. There is no requirement of immediate precedence and no constraint on immediate succession either. An MDP representation, on the other hand, would lay out exact precedence and succession orderings of methods within a path in the MDP tree.

3. TÆMS can be thought of as a compact representation of a class of MDP problems. It implicitly describes the enumerated search space which is explicitly described by the MDP.

The translation process of a TÆMS task structure to a MDP involves following a procedure which lays out each possible execution path for achieving the high level goal. The algorithm for expanding the compact TÆMS representation to an elaborate MDP representation is described in the next section.

5 The Translation

The MDP translation is a procedure which allows for the transformation of a TÆMS task structure T to the corresponding MDP M. The state in the MDP representation is a vector which represents the methods that have been executed in order to reach that state along with their execution characteristics. The MDP action is the execution of a particular method. MDP actions have outcomes and each outcome is characterized by a 3-tuple consisting of discrete quality, cost and duration values obtained from the expected performance distribution of the MDP action. The rewards are computed only for the terminal states, i.e. the intermediate states have null rewards. The reward is computed by applying a complex criteria evaluation function[2] of the quality, cost and duration values obtained by the terminal state. Value iteration is the dynamic programming algorithm used to compute the optimal policy. In theory, value iteration requires an infinite number of iterations to converge to the optimal policy. In practice, however we stop once the value function changes by only a small amount in a sweep. The following is the algorithm for the translation process.

Let TG be the top level goal in T and let METHODS be the set of primitive actions in T.

```
1. Initialize MDP with state $s$;
2. *Translate(s)*
    (a) Identify the set of actions(subset of METHODS) which are possible
        from $s$.
    (b) Iterate over each action
        i. If action is not TERMINATE
            A. Expand each outcome(characterized by discrete quality, cost,
               duration values).
```

[2]the same multi-dimensional evaluation mechanism used by DTC is used here to ensure a fair comparison.

```
            B. Determine if outcome can lead to a new state while adhering
               to the criteria constraints.
            C. if new state $s_{prime}$ is reached,
               $Translate(s_{prime})$
        ii. Else if action is TERMINATE, set reward of terminating state to
            be a function of the quality, cost and duration values of the state.
3. ValueIteration(StateSet);
4. Return optimal policy.
```

To make this discussion on the translation process more concrete, we will apply a few iterations of the algorithm on the example discussed in this paper. Upon translation, the corresponding MDP has 49 states. Figure 4 describes the translation process in progress. The input to the algorithm is the task structure in Figure 1 described in textual format. The start state *SO* is initialized. The PossibleActionSet for state *SO* is {*SearchAdobeURL*, *UserBenchMarks*, *FindUserReviews*}. *ApplyNLP* is not a valid action since it has an inactive incoming enables from *FindUserReviews.* For each action in PossibleActionSet, we consider the outcomes of each of the action starting with *SearchAdobeURL* which has 4 outcomes resulting in the states *S1*, *S7*, *S13*, *S21* respectively. The outcome resulting in state *S1* has a quality of 0.5, cost of 8.0 and duration of 3.0 and occurs with a frequency of 16%. We now determine the PossibleActionSet for state *S1.* The set is {*UserBenchMarks*, *FindUserReviews*, *Terminate*}. *UserBenchMarks* has 3 possible outcomes resulting in the states *S2*, *S3* and *S4.* The PossibleActionSet for state *S2* is {*Terminate* and *FindUserReviews*}. We exit from the current loop when a *Terminate* action is encountered and also exit from current loop if a deadline is crossed as is the case for both outcomes of *FindUserReviews.*

The optimal policy for the above problem is shown in Figure 5. The policy suggests the method sequence {*FindUserReviews*, *UserBenchMarks*, *ApplyNLP*} (A2,A1,A3) as the best schedule when method *FindUserReviews* achieves non-zero quality and {*FindUserReviews*, *SearchAdobeURL*} (A2,B) would be an alternate schedule in the event of *FindUserReviews*'s failure to achieve quality.

As mentioned earlier, the MDP state space of task structures modeling real-world applications undergoes a combinatorial explosion. For task structures with 30-40 primitive actions, the DTC scheduler takes generally less than ten seconds to execute on a mid-range Pentium III running Linux [17]. The contingency enhanced version of DTC requires more time and resources, however, because it generally explores only a portion of the possible solution

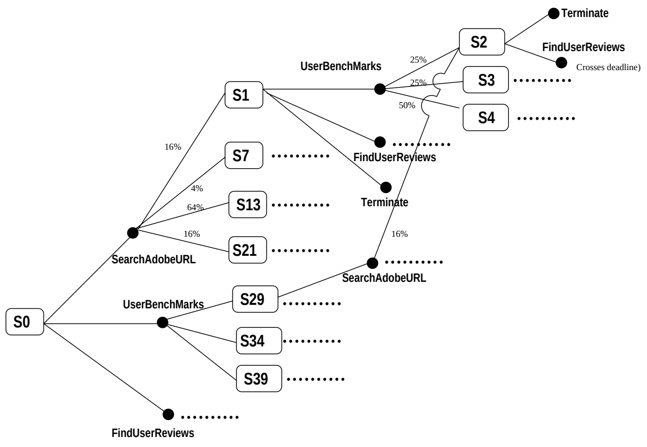

Figure 4: Translation process from TAEMS task structure to Markov Decision Process for Gather-Review-Information-on-AdobePhotoshop example.

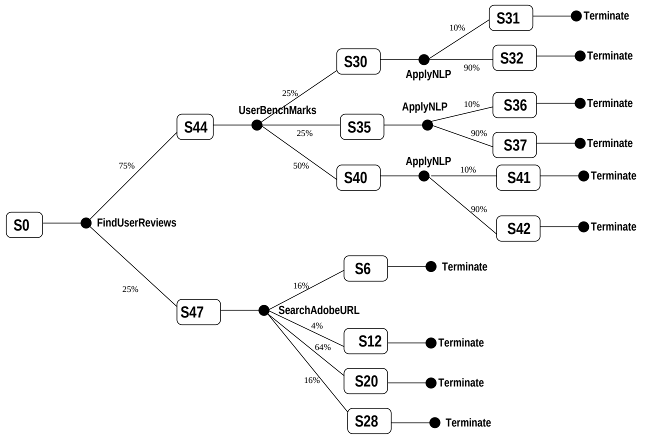

Figure 5: Optimal policy for Gather-Review-Information-on-AdobePhotoshop

space, it is able to schedule problems too large for the optimal MDP. Hence, this approach has mainly been researched to serve as an off-line theoretical benchmark to evaluate our real-time heuristic schedulers.

6 Experimental Results

Effective contingency planning is a complex process. It involves taking into account a number of factors, including task relationships, deadlines, the availability of alternatives, and client design criteria (i.e., quality, cost, duration, and certainty trade-offs). In this section, we evaluate the performance of the contingency analysis tools by comparing them to the standard Design-to-Criteria scheduler as well as to the MDP-based optimal meta-controller. It is possible to characterize the types of task structures that are amenable to contingency analysis, i.e., those for which analysis of recovery options is beneficial from a cost/benefit perspective. As part of the evaluation process, we have partially determined the characteristics of task structures and design criteria that indicate a problem instance for which contingency planning is advantageous.The general characteristics include:

1. Methods in task structures should have a possibility of failure in their distribution. Contingency analysis is worth the associated computational overhead only if there is a possibility of failure of the current schedule to meet the high-level goal due to individual method failure. If the performance of the best schedule is deterministic, contingency analysis is dispensable.
2. Task structures should contain alternate paths. The absence of of possible recovery paths in the face of failure also makes contingency analysis dispensable.
3. A possibility of moving failure methods forward (absence of associated hard non-local effects) would further the potential of contingency analysis, i.e., structures in which there is some flexibility in terms of method placement within a schedule. If methods have strong precedence and succession constraints by way of enables non-local effects, and the failure points are in the latter portion of the schedule, then there is little possibility of finding good recovery options for failure within the resource constraints.
4. Dependence of methods with good average performance on critical methods (enables non-local effect from a critical method to a non-critical method). Extensive contingency analysis is required only if the critical regions affect the rest of the schedule significantly. Otherwise, a cheap local fix by replacing the critical method by a more stable method or just adding a redundant method within the criteria requirements is a better choice than contingency analysis.

TS #	Normal		Contingency		MDP		p1	p2
	A.Q.	σ	A.Q.	σ	A.Q.	σ		
1	145.80	8.19	160.2	33.09	166.20	32.31	0.00	0.20
2	73.6	19.77	91.9	16.06	90.40	16.93	0.00	0.52
3	73.4	21.75	88.3	24.21	86.80	24.57	0.00	0.66
4	309.0	281.63	342.20	204.20	391.40	278.50	0.34	0.16
5	119.99	47.15	131.58	39.97	122.59	50.29	0.06	0.16
Col. #	1	2	3	4	5	6	7	8

Table 1: Experimental Results:*Normal A.Q.* is average quality of ELB selected schedule; *Normal* σ is the standard deviation ELB selected schedule's quality; *Contingency A.Q.* is average quality of AEB selected schedule; *Contingency* σ is the standard deviation AEB selected schedule's quality;*MDP A.Q.* is average quality of optimal meta-control policy; *MDP* σ is the standard deviation of the optimal policy's quality; *p1* is the p-value for the t-test comparing *Normal A.Q.* and *Contingency A.Q.*; *p2* is the p-value for the t-test comparing *Contingency A.Q.* and *MDP A.Q.*

The following are the design criteria characteristics 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. The hard deadline and other such hard resource constraints voids the possibility of simply rescheduling at failure points and instead requires off-line contingency analysis.
2. The deadline should also provide enough time for contingency analysis, if the scheduling cost is factored into the equation. Regardless, the deadline must provide sufficient time for recovery options to be deployed otherwise the existence of such options is meaningless. In these cases, the contingency analysis tools must resort to the same single-pass execution view that is used in the main Design-to-Criteria scheduler.

The first stage of the performance evaluation compares the contingency-enhanced DTC scheduler to the normal DTC scheduler. Comparison is done by examining the Expected Lower Bound (standard scheduler metric) and the Approximate Expected Bound (contingency analysis metric) and comparing schedules selected on the basis of these metrics to the actual results obtained by executing the schedules in a simulation environment. The second stage of the evaluation compares the performance of the schedule with highest Approximate Expected Bound to that of the optimal policy prescribed by the MDP-based meta-controller.

The experiments in this section were conducted by randomly generating task structures while varying some of the above mentioned characteristics. This produced structures amenable to contingency analysis and were used to seed the search for interesting test cases. Since method failure is a crucial factor for the contingency analysis argument, the generation of task structures was designed to concentrate on the variance of two factors, namely, the effects of failure location and failure intensity (probability of failure) within a task structure. Failure location refers to the position of critical method(s) in a task structure and hence in the schedule. Failure intensity refers to the probability of a method failing. Three different classifications of failure location are used in the experiments: early, medium, and late. This is accomplished via non-local effects and sequencing-related quality accumulation functions that force particular actions to be carried out at particular points in any schedule including the actions. Similarly, three different settings for failure intensity are used in the experiments, namely, low, medium and high where low is 1%-10% probability of failure, medium is 11%-40%, and high is 41%-90%. Ten randomly generated task structure classes were modified to varying degrees with respect to these two factors, producing in 86 task structure instances.

The design criteria in these experiments is to maximize quality given a hard deadline on the overall schedule. This simple design criteria setting is one that lends itself to contingency analysis as the existence of a hard deadline (in contrast to a soft preference, e.g., soft deadline) may preclude recovery via rescheduling in certain circumstances. Because of the hard deadline, a poorly chosen initial schedule may not leave time for the deployment of recovery options and thus the normal Design-to-Criteria scheduler may fail to produce results in situations where contingency analysis has planned for the recovery scenario and chosen an initial schedule accordingly

The results for the experiments are shown in Table 1. For each task structure instance, 100 simulated executions were performed using the schedule with the highest Expected Lower Bound(ELB), the schedule having the highest Approximate Expected Bound(AEB) and the optimal meta-control policy, i.e., the best schedule selected by the Design-to-Criteria scheduler was executed a 100 times, the best schedule selected by (or generated by, in the case of method movement) contingency analysis was executed 100 times and the optimal control policy for the task structure was executed a 100 times.

Each row in the table represents a specific task structure. The null hypothesis of equivalence could not be rejected at the .05 level via a one-tailed t-test for 83 of the total 86 task structures. In other words, the results produced via AEB are not statistically significantly different from the results produced by the ELB for 83 of the total 86 task structures. Rows four and five show the execution results on two such task structures where all three algorithms have the same level of performance. Generally these are instances where the schedule selected by all three methodologies are the same, indicating a lack of many appealing options that may serve to lure the standard Design-to-Criteria scheduler away from the schedule that also happens to have recovery options associated with it. The elimination of many of the task structures is evidence that it is difficult to pre-determine whether contingency planning is expedient for a certain task structure.

Rows one, two and three represent the three task structures for which the the null hypothesis of equivalence was rejected at the .05 level. These are the task structures that led to schedules for the ELB case and the AEB case that produced execution results that are statistically significantly different.

Columns one, three and five show the average quality that was produced by the ELB selected, AEB selected and optimal control policy respectively. Columns two, four and six show the corresponding standard deviations. In other words, the best schedule per the ELB metric was selected and executed in an unbiased simulation environment, when failure occurred the scheduler was reinvoked. The resultant quality was measured and recorded and the experiment repeated 100 times. The same procedure was done for the ELB selected schedule, though when rescheduling occurred, the scheduler and the contingency analysis tools were reinvoked. In the case of the optimal control policy, the policy was executed a 100 times and for each state that was reached, the best action prescribed by the policy was executed and the action outcome of the action dictated the next reachable state. For rows one, two and five the average quality of the MDP-based policy is lower than the average quality of the schedule with highest AEB. On examining the data in detail, we found that this was essentially due to the skew of the random number generator. Increasing the number of runs to 500 resulted in the average quality of the optimal policy to be greater than or equal to that of the schedule with highest AEB, which is as expected.

Columns seven, titled *p1*, shows the p-value for the t-test comparing the ELB selected schedule and the AEB selected schedule. For the first three rows, the p-value is less than or equal to 0.05. Hence for these three task structures, performance of contingency-enhanced scheduler is statistically significantly different from the performance of the normal scheduler. For rows four and five, the p-value is greater than 0.05, hence the null hypothesis could not rejected. Hence there is no statistically significant difference in the performance achieved by applying contingency analysis on these task structures.

Column eight, titled *p2* is th p-value for the t-test comparing the AEB selected schedule to the optimal meta-control policy. For all five task structures, the p-values were greater than 0.05. Hence for these five task structures, the performance of the AEB selected schedule is not statistically significantly different from the performance of the optimal controller. One might infer that in these specific cases the AEB selected schedule is near optimal or closely approximates optimal.

7 Conclusions and Future Work

Constructing an optimal controller is an NP-hard problem and makes this approach infeasible. We have hence constructed a heuristic scheduler which satisfices all aspects of problem solving. We then augmented the scheduler with contingency analysis tools to effectively deal with and reason about mission-critical situations [12] . In this paper, we identify the characteristics of problem instances for which contingency analysis is amenable. We then evaluate the performance of the schedule produced by contingency analysis to that of an optimal controller.

Our preliminary results in Section 6 show that for a certain sub-class of task structures, namely those task structures which have the properties of mission-critical systems, applying contingency analysis is advantageous and leads to performance that is not statistically significantly different from that of the optimal policy. These task structures have critical task execution regions and are constrained by hard deadlines and mid-stream schedule failure could lead to catastrophic system-wide failure.

Further experiments comparing the normal DTC scheduler to the contingency-enhanced scheduler are discussed in [15]. Our current research focuses on extending our experimental evaluation and pre-classifying task structures that are potentially amenable to contingency analysis. We are looking into applying formal search techniques to construct the policy. This approach would allow the use of the vast amount of legacy algorithms to handle approximations and provide guarantees of optimality.

In the current implementation, the cost of rescheduling is considered to be a small overhead and the costs associated with rescheduling are effectually ignored. We are exploring techniques to handle situations where non-trivial costs are associated to rescheduling.

8 Acknowledgments

We would like to thank Professor Shlomo Zilberstein for his contribution in the presentation of the contingency analysis algorithm as well as the discussions on computing rewards for the MDP's. We also thank the anonymous reviewers and David Fisher for their useful comments.

References

[1] A.G. Barto, S. J. Bradtke, and S.P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, 1995.

[2] C. Boutilier, T. Dean, and S. Hanks. Planning under uncertainty: Structural assumptions and computational leverage. In *Proceedings of 3rd European Workshop on Planning (EWSP'95)*, 1995.

[3] Thomas Dean, Leslie Kaelbling, Jak Kirman, and Ann Nicholson. Planning under time constraints in stochastic domains. *Artificial Intelligence*, 76(1-2):35–74, 1995.

[4] R. Dearden and C. Boutilier. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence*, 89:219–283, 1997.

[5] Keith S. Decker and Victor R. Lesser. Coordination assistance for mixed human and computational agent systems. In *Proceedings of Concurrent Engineering 95*, pages 337–348, McLean, VA, 1995. Concurrent Technologies Corp. Also available as UMASS CS TR-95-31.

[6] D. Draper, S. Hanks, and D. Weld. 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, 1994.

[7] Alan Garvey and Victor Lesser. Design-to-time scheduling with uncertainty. CS Technical Report 95–03, University of Massachusetts, 1995.

[8] P. Haddaway and S. Hanks. Utility models for goal-directed decision-theoretic planners. *Computer Intelligence*, 14(3), 1998.

[9] N. Kushmerick, S. Hanks, and D. Weld. An algorithm for probabilistic planning. In *Proceedings of the Eleventh National Conference on Artificial Intelligence,*, 1994.

[10] Victor Lesser, Michael Atighetchi, Bryan Horling, Brett Benyo, Anita Raja, Regis Vincent, Thomas Wagner, Ping Xuan, and Shelley XQ. Zhang. A Multi-Agent System for Intelligent Environment Control. In *Proceedings of the Third International Conference on Autonomous Agents (Agents99)*, 1999.

[11] Victor Lesser, Bryan Horling, Frank Klassner, Anita Raja, Thomas Wagner, and Shelley XQ. Zhang. BIG: A resource-bounded information gathering agent. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, July 1998. See also UMass CS Technical Reports 98-03 and 97-34.

[12] David J. Musliner. Plan Execution in Mission-Critical Domains. In *Working Notes of the AAAI Fall Symposium on Plan Execution - Problems and Issues*, 1996.

[13] T. Oates, M. V. Nagendra Prasad, and V. R. Lesser. Cooperative Information Gathering: A Distributed Problem Solving Approach. Computer Science Technical Report 94–66, University of Massachusetts, 1994. Journal of Software Engineering, Special Issue on Developing Agent Based Systems, 1997.

[14] N. Onder and M. Pollack. Contingency selection in plan generation. In *Proceedings of the Fourth European Conference on Planning*, 1997.

[15] Anita Raja, Thomas Wagner, and Victor Lesser. Reasoning anout Uncertainty in Design-to-Criteria Scheduling. Computer Science Technical Report TR-99-27, University of Massachusetts at Amherst, March 2000. To appear in the 2000 AAAI Spring Symposium on Real-Time Systems.

[16] Thomas Wagner, Alan Garvey, and Victor Lesser. Criteria-Directed Heuristic Task Scheduling. *International Journal of Approximate Reasoning, Special Issue on Scheduling*, 19(1-2):91–118, 1998. A version also available as UMASS CS TR-97-59.

[17] Thomas Wagner and Victor Lesser. Design-to-Criteria Scheduling: Real-Time Agent Control. Computer Science Technical Report TR-99-58, University of Massachusetts at Amherst, March 2000. To appear in the 2000 AAAI Spring Symposium on Real-Time Autonomous Systems.

[18] Thomas Wagner, Anita Raja, and Victor Lesser. Modeling Uncertainty and its Implications to Design-to-Criteria Scheduling. *Under review, also available as UMASS Department of Computer Science Technical Report TR-98-54*, 1999.

[19] Shlomo Zilberstein and Abdel illah Mouaddib. Reactive Control of Dynamic Progressive Processing. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999.