Design-to-Criteria Scheduling: Real-Time Agent Control *†

Thomas Wagner
Computer Science Department
University of Maine
Orono, ME 04469
wagner@umcs.maine.edu

Victor Lesser
Computer Science Department
University of Massachusetts
Amherst, MA 01003
lesser@cs.umass.edu

Abstract

Design-to-Criteria builds custom schedules for agents that meet hard temporal constraints, hard resource constraints, and soft constraints stemming from soft task interactions or soft commitments made with other agents. Design-to-Criteria is designed specifically for online application – it copes with exponential combinatorics to produce these custom schedules in a resource bounded fashion. This enables agents to respond to changes in problem solving or the environment as they arise.

Introduction

Complex autonomous agents operating in open, dynamic environments must be able to address deadlines and resource limitations in their problem solving. This is partly due to characteristics of the environment, and partly due to the complexity of the applications typically handled by software agents in our research. In open environments, requests for service can arrive at the local agent at any time, thus making it difficult to fully plan or predict the agent's future workload. In dynamic environments, assumptions made when planning may change, or unpredicted failures may occur [1]. In most real applications, deadlines or other time constraints are present on the agent's problem solving [16, 8]. For example, in an anti-submarine warfare information gathering application [3], there is a deadline by which the mission planners require the information. Resource limitations may also stem from agents having multiple different tasks to perform and having bounded resources in which to perform them. Temporal constraints may also originate with agent interactions – in general, in order for agent β to coordinate with agent α, the agents require mutual temporal information so that they can plan downstream from the interaction.

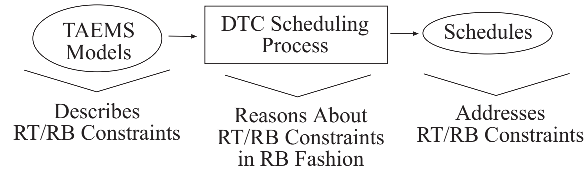
TAEMS Models
DTC Scheduling Process
Schedules
Describes RT/RB Constraints
Reasons About RT/RB Constraints in RB Fashion
Addresses RT/RB Constraints

Figure 1: Modeling and Online Scheduling for Real Time and Resource Boundedness

In this paper, we focus on the issue of resource bounded agent control. We use the term *resource bounded* to denote the existence of deadlines and of other constraints like cost limitations or application specific resource limitations (e.g., limited network bandwidth). Where it is important to differentiate hard and soft deadlines from these other constraints, we refer to them explicitly.

For agents to adapt rationally to their changing problem solving context, which includes changes in the environment [2] and changes to the set of duties for the agent to perform, they must be able to:

1. Represent or model the time and resource constraints of the situation and how such constraints impact their problem solving. We believe this must be done in a quantified fashion as different constraints have different degrees of effect on problem solving.

2. Plan explicitly to address the resource limitations. In our work, this may imply performing a different set of tasks, using alternate solution methods, or trading-off different resources (or quality), depending on what is available.

[*] UMass Computer Science Technical Report 1999-58. A version appears in the Proceedings of 2000 AAAI Spring Symposium on Autonomous Real-Time Systems.

[†] This material is based upon work supported by the National Science Foundation under Grant No. IIS-9812755, by the Department of the Navy and Office of the Chief of Naval Research, under Grant No. N00014-97-1-0591, and by the Defense Advanced Research Projects Agency (DARPA) and AFRL under Grant No. F30602-99-2-0525. This work is also funded by the National Defense Center of Excellence for Research in Ocean Sciences (CEROS). CEROS is a part of the Natural Energy Laboratory of Hawaii Authority (NELHA), an agency of the Department of Business, Economic Development & Tourism, State of Hawaii. CEROS is funded by DARPA through grants to and agreements with NELHA. The content of the information does not necessarily reflect the position or the policy of the Government, NSF, ONR, DARPA, AFRL, or CEROS, and no official endorsement should be inferred.

[1] This differs from states that are explicitly recognized and *planned for* [1] as software agents may be required to perform a different set of tasks, as well as having to react to changes in the environment.

[2] Including resources uncontrollably becoming more or less constrained. For example, network latency increasing due to some activity other than the agent's problem solving.

3. Perform this planning online – in the general case, this implies coping with exponential combinatorics online in soft real time.

While the first two requirements obviously follow from the domain, the third requirement is less obvious. Agents must be able to perform real time control problem solving online because of the dynamics of the environment. If it is difficult to predict the future and there is a possibility of failure, or new tasks arriving, agents will, by necessity, have to react to new information and replan online.

The Design-to-Criteria (DTC) agent scheduler and the TÆMS task modeling framework are our tools for addressing these requirements and achieving resource-bounded agent control (Figure 1). TÆMS provides agents with the framework to represent and reason about their problem solving process from a quantified perspective, including modeling of interactions between tasks and resource consumption properties. Design-to-Criteria performs analysis of the processes (modeled in TÆMS) and decides on an appropriate course of action for the agent given its temporal and resource constraints. Design-to-Criteria both produces resource-aware schedules for the agent, and, does this reasoning process online in a resource bounded fashion.

While the output of Design-to-Criteria is real time in the sense that the schedules address hard and soft deadlines, and resource constraints, the schedules are not hard real time and are not fault tolerant in the sense that they may contain uncertainty and known potential failure points. Because DTC is applied in domains where failure is expected, and modeled, and rescheduling is expected, it may often be prudent to choose a schedule that contains some probability of failure, but, also some probability of higher returns. The issue of uncertainty, and its role in addressing hard deadlines, is covered in greater detail later. For situations in which a mid-stream schedule failure leads to catastrophic system-wide failure, we have developed an offline variant of DTC that uses contingency analysis [17, 23] to explore and evaluate recovery options from possible failure points.

This paper is organized as follows: in Section we present TÆMS and describe its role in our domain independent approach to agent control. In Section we describe how DTC reasons about the agent's context and makes control decisions to produce resource bounded schedules. In Section , DTC's approximate online solution strategy is presented and in Section we discuss limitations, open questions, and future work.

TÆMS Task Models

TÆMS (Task Analysis, Environment Modeling, and Simulation) [6] is a domain independent task modeling framework used to describe and reason about complex problem solving processes. TÆMS models are used in multi-agent coordination research [24, 11] and are being used in many other research projects, including: cooperative-information-gathering [14], hospital patient scheduling [5], intelligent environments [13], coordination of software process [12], and others [20]. Typically, in our domain-independent agent architecture, a domain-specific problem solver or planner translates its problem solving options 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 scheduler. The control problem solvers then decide on an appropriate course of action for the agent, possibly by coordinating and communicating with other agents (that also utilize the same control technologies).

TÆMS models are hierarchical abstractions of problem solving processes that describe alternative ways of accomplishing a desired goal; they represent major tasks and major decision points, interactions between tasks, and resource constraints but they do not describe the intricate details of each primitive action. All primitive actions in TÆMS, called *methods*, are statistically characterized via discrete probability distributions in three dimensions: quality, cost and duration. Quality is a deliberately abstract domain-independent concept that describes the contribution of a particular action to overall problem solving. 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. Uncertainty in each of these dimensions is implicit in the performance characterization – thus agents can reason about the certainty of particular actions as well as their quality, cost, and duration trade-offs. The uncertainty representation is also applied to task interactions like *enablement, facilitation* and *hindering* effects, [3] e.g., "10% of the time facilitation will increase the quality by 5% and 90% of the time it will increase the quality by 8%."

The quantification of actions and interactions in TÆMS is not regarded as a perfect science. Task structure programmers or problem solver generators *estimate* the performance characteristics of primitive actions. These estimates can be refined over time through learning and reasoners typically replan and reschedule when unexpected events occur.

To illustrate, consider Figure 2, which is a conceptual, simplified sub-graph of a task structure emitted by the BIG [14] resource bounded information gathering agent; it describes a portion of the information gathering process. The top-level task is to construct product models of retail PC systems. It has two subtasks, *Get-Basic* and *Gather-Reviews*, both of which are decomposed into actions, that are described in terms of their expected quality, cost, and duration. The *enables* arc between *Get-Basic* and *Gather* is a non-local-effect (NLE) or task interaction; it models the fact that the review gathering actions need the names of products in order to gather reviews for them. Other task interactions modeled in TÆMS include: *enablement*, *facilitation*, *hindering*, *bounded facilitation*, *disablement*, *consumes-resource* and *limited-by-resource*. Task interactions are important to scheduling because they denote points at which a task may be affected, either positively or nega-

[3] Facilitation and hindering task interactions model soft relationships in which a result produced by some task may be beneficial or harmful to another task. In the case of facilitation, the existence of the result generally increases the quality of the recipient task or reduces its cost or duration.

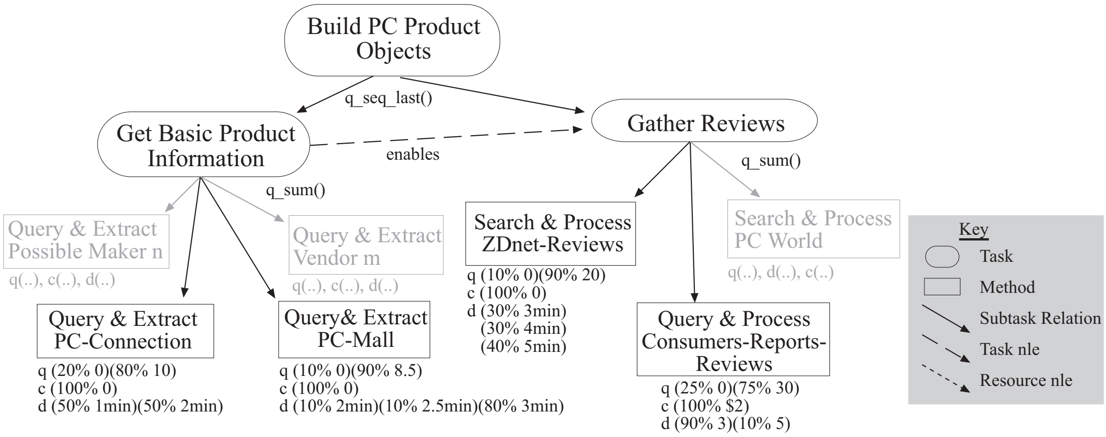

Figure 2: Conceptual Information Gathering Task Structure of the BIG Agent

tively, by an outcome elsewhere in the task structure (or at another agent).

Returning to the example, *Get-Basic* has two actions, joined under the *sum()* quality-accumulation-function (*QAF*), which defines how performing the subtasks relate to performing the parent task. In this case, either action or both may be employed to achieve *Get-Basic*. The same is true for *Gather-Reviews*. The QAF for *Build-PC-Product-Objects* is a *seq_last()* which indicates that the two subtasks must be performed, in order, and that the quality of *Build-PC-Product-Objects* is determined by the resultant quality of *Gather-Reviews*. There are nine alternative ways to achieve the top-level goal in this particular sub-structure.[4] In general, a TÆMS task structure represents a family of plans, rather than a single plan, where the different paths through the network exhibit different statistical characteristics or trade-offs. The process of deciding which tasks/actions to perform is thus an *optimization* problem rather than a *satisfaction* problem.

TÆMS also supports modeling of tasks that arrive at particular points in time, parallelism, individual deadlines on tasks, earliest start times for tasks, and non-local tasks (those belonging to other agents). In the development of TÆMS there has been a constant tension between representational power and the combinatorics inherent in working with the structure. The result is a model that is non-trivial to process, coordinate, and schedule in any optimal sense (in the general case), but also one that lends itself to flexible and approximate processing strategies. This element of choice and flexibility is leveraged both in designing resource-bounded schedules for agents and in performing online scheduling in a resource bounded fashion.

Modeling and Reasoning about Temporal and Resource Constraints

TÆMS tasks may have both soft and hard constraints that must be considered when scheduling. In terms of hard temporal constraints, any TÆMS task may have a hard deadline, by which some quality must be produced (or it is considered a failure), as well as an earliest-start-time, before which the task may not be performed (or zero quality will result). These hard constraints may also be caused by hard commitments[5] made with other agents or hard delays between task interactions. The constraints may also be inherited from nodes higher in the structure – thus a client may specify a hard deadline on the *Build-PC* task that applies to all subtasks, or a deadline may be specified on the process of *Gathering-Reviews*. If multiple temporal constraints are present, the tightest or most conservative interpretation applies.

Recall that actions in TÆMS are characterized using discrete probability distributions. Because durations may be uncertain, and because actions are sequenced in a linear fashion,[6] the implication of duration uncertainty is that there is generally uncertainty in both the start and finish times of tasks – even tasks that do not have duration uncertainty of their own. When each TÆMS action is added to a schedule, or considered for a particular schedule point, a data structure called a *schedule element* is created and the start, finish, and duration distributions for the schedule element are computed as a function of the characteristics of the previous schedule element and the action being scheduled. The constraints associated with the action (and higher level task) are then examined and compared to the characteristics that will result if the action is performed at the "current" time or point in the schedule.

One approach for determining whether or not a given action will violate a hard deadline, for example, is to look at some single statistic (median, mean, max, min) of the

[4] While it might appear per the *seq_last()* QAF that there are only two possible resultant quality distributions, the enables interaction between *Build* and *Gather* affects the possible quality values for *Gather*.

[5] In contrast to commitments that are soft or relaxable, possibly through a decommitment penalty mechanism.

[6] While DTC supports scheduling of specialized parallel activities, even when activities are scheduled in parallel, they may inherit uncertainty from prior activities.

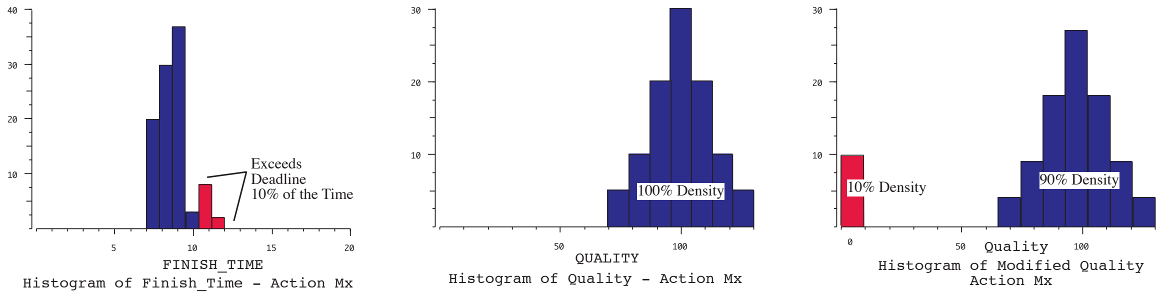

Figure 3: Reflecting Probability of Missing Deadline in Method Quality

action and to compare that statistic to the deadline, e.g., $if\ (mean(finish\_time(action_x)) > hard\_deadline(action_x))$ $then\ violated$. This approach is used during some of the approximation processes of the scheduling algorithm. Another reasonable approach is to compute the probability that the action will violate its hard constraint and compare the probability to a predetermined threshold, e.g., $P = Pr(finish\_time(action_x) > hard\_deadline(action_x))$. $if\ P > Threshold_P\ then\ violated$.

However, TÆMS provides us with a better tool for reasoning about constraint violation. Because zero quality reflects failure, and in TÆMS an action that violates its hard deadline produces zero quality, we can reason about the probability that a given action violates its hard deadline simply by reflecting said probability in the quality distribution of the action and then treating it like any other TÆMS action.[7] Enforcing hard deadline constraints on the agent's entire process (analogous to imposing a deadline on the task structure root) is handled in the same way. For example, as shown in Figure 3, if M_x has a 10% chance of exceeding its deadline (and thus failing), the densities of all the members of its quality distribution are multiplied by 90% (thus re-weighting the entire distribution) and a new density / value pair is added to the distribution to reflect the 10% chance of returning a result after the deadline. The leftmost histogram describes M_x's expected finish time, the middle histogram describes M_x's unmodified quality distribution, and the rightmost figure shows the modified quality distribution after re-weighting and merging with the new $(10\%,\ 0\ quality)$ pair. Through this solution approach, the scheduler may actually select a course of action for the agent that has some probability of failure, however, the probability of failure is reflected directly in solution quality so that if the risk is not worthwhile (relative to the other solution paths available to the agent) it will not be taken. In other words, a path containing a possible deadline violation will only be chosen if it has a higher quality than the other solutions on an expected value basis.

On the surface, this model is not appropriate for hard real time applications in which the failure of the action results in no solution for the agent. However, if this is the case, the action will serve a key role in the task structure or will interact with (e.g., enable) other actions in the structure and thus the failure will result in the quality of the affected actions also being decreased and further lower solution quality. The view presented here, if modeled appropriately, gives the scheduler a very powerful tool for reasoning about the implications of possible failures and their impact on overall problem solving.

In addition to hard temporal constraints, TÆMS also models hard resource constraints. For example, a given task may require the use of a network connection and without this connection, the task may produce zero quality (fail). In TÆMS, the effects of resource constraints are modeled using a *limits* NLE from the resource to the task where the NLE describes a multiplier relationship between the resource and the task. For example, running out of a resource may cause the task to take 1.5 times as long to execute, or it may cause the quality to decrease by 50%, or it may cause the cost to increase, or it may simply cause failure. As with violating a hard temporal constraint, if a resource constraint causes action failure, it is reflected in the quality of the action and any actions or tasks that are acted-upon (e.g., by an enables from the affected action) will also have their qualities adjusted to reflect the effects of the resource problem.

Soft constraints in TÆMS take the form of soft commitments made with other agents and soft interactions between tasks. For example, if task α *facilitates* β, performing α before β will positively affect β, possibly by shorting β's duration, but the facilitation does not need to be leveraged to perform either task. When scheduling for soft constraints associated with actions, the scheduler attempts to utilize them when possible (or avoid the in the case of a soft negative interaction, e.g., *hinders*). However, whenever a soft constraint is violated, either on the positive or negative side, the quality distributions of the involved actions are modified to reflect the situation and thus the scheduler can again reason directly about the impact of constraint violation on the agent's process.

The scheduler also supports soft constraints on overall problem solving. In addition to setting hard temporal constraints, the scheduler client may specify an overall soft deadline, soft cost limit, or soft quality requirement. These soft constraints are members of a package of client preferences called *design criteria* that describes for the scheduler the client's objective function. The scheduler then works to produce a schedule (or set of schedules) to suit the client's needs. The criteria mechanism is soft because, due to the

[7] Professor Alan Garvey, developer of a forerunner to Design-to-Criteria, Design-to-Time, first used a similar technique in Design-to-Time. The technique presented here was developed independently in the DTC research.

Schedule A - Client has no resource limitations, maximize quality.

```
Query-and-Extract-PC-Connection | Query-and-Extract-PC-Mall | Query-and-Process-ZDnet | Query-and-Process-Consumers

Quality distribution: (0.04 0.00)(0.22 20.00)(0.07 30.00)(0.66 50.00)
Expected value:  39.69
Probability q or greater:  0.66
Cost distribution: (1.00 2.00)
Expected value:  2.00
Probability c or lower:  1.00
Finish time distribution: (0.02 9.00)(0.14 10.00)(0.03 10.50)(0.25 11.00)(0.03 11.50)(0.00 11.65)(0.30 12.00)(0.18 13.00)
            (0.03 14.00)(0.02 15.00)
Expected value:  11.65
Probability d or lower:  0.47
```

Schedule B - Client interested in a free solution.

```
Q&E-PC-Connection | Q&E-PC-Mall | Q&P-ZDnet

Quality distribution: (0.12 0.00)(0.88 20.00)
Expected value:  17.64
Probability q or greater:  0.88
Cost distribution: (1.00 0.00)
Expected value:  0.00
Probability c or lower:  1.00
Finish time distribution:  (0.02 6.00)(0.02 6.50)(0.15 7.00)(0.03 7.50)
      (0.28 8.00)(0.04 8.50)(0.30 9.00)(0.02 9.50)(0.16 10.00)
Expected value:  8.45
Probability d or lower:  0.49
```

Schedule C - Maximize quality while meeting hard deadline of 6min.

```
Q&E-PC-Mall | Q&P-Consumers

Quality distribution: (0.39 0.00)(0.61 30.00)
Expected value:  18.23
Probability q or greater:  0.61
Cost distribution: (1.00 2.00)
Expected value:  2.00
Probability c or lower:  1.00
Finish time distribution: (0.09 5.00)(0.09 5.50)(0.72 6.00)(0.01 7.00)
            (0.01 7.50)(0.08 8.00)
Expected value:  6.05
Probability d or lower:  0.90
```

Figure 4: Different Schedules for Different Clients

combinatorics of reasoning about TÆMS task structures, it is often difficult to predict what types of solutions are possible. Instead, the client describes the desired solution space in terms of relaxable, relative, design criteria (in quality, cost, duration, uncertainty in each dimension, and limits and thresholds on these) and the scheduler makes trade-off decisions as needed to best address the client's needs. The criteria metaphor is based on importance sliders for quality, cost, duration, limits and thresholds on these, and certain in each of these dimensions. The metaphor, the formal mathematics of the criteria mechanism, and the scheduler's trade-off computations have been fully documented in [22, 21].

Let us revisit BIG's process, shown in Figure 2, and illustrate DTC's creation of custom, resource bounded, schedules and the role of task interaction in modeling the effects of failure. Even this simple task structure gives DTC room to adapt BIG's problem solving. Figure 4 shows three different schedules constructed for different BIG clients that have different objectives. For brevity, the detailed distributions associated with each action are omitted, however, the aggregate schedule statistics are shown. Schedule A is constructed for a client that has both time and financial resources – he or she is simply interested in maximizing overall solution quality. Schedule B is constructed for a client that wants a free solution. Schedule C meets the needs of a client interested in maximizing quality while meeting a hard deadline of 6 minutes. Note that schedule C is actually preferred over a schedule that includes action *Query-and-Extract-PC-Connection* even though said action has a higher expected quality than *Query-and-Extract-PC-Mall*. This is because the *PC-Connection* action also has a higher probability of failure. Because of the enables NLE from the task of getting product information to retrieving reviews, this higher probability of failure also impacts the probability of being able to query the Consumer's site for a review. Thus, though the local choice would be to prefer *PC-Connection* over *PC-Mall* for this criteria, the aggregate effects lead to a different decision. Note also that schedule C also exceeds its deadline 10% of the time. The deadline over-run and the enablement from *PC-Mall* contribute to the probability of failure exhibited by the schedule (probability of returning a zero quality result), i.e., Consumer's fails 25% of the time without considering these other constraints. When considering the other constraints, probability of failure is: (((25% * .90) + 10%) * .90) + 10% = 39.25%.

Online Scheduling - Coping with Exponential Combinatorics

As TÆMS task structures model a family of plans, the DTC scheduling problem has conceptually certain characteristics in common with planning and certain characteristics of more traditional scheduling problems, and it suffers from pronounced combinatorics on both fronts. The scheduler's function is to read as input a TÆMS task structure (or a set of task structures) and to 1) decide which set of tasks to perform, 2) decide in what sequence the tasks should be performed, 3) to perform the first two functions so as to address hard constraints and balance the soft criteria as specified by the client,[8] and 4) to do this computation in soft real time (or interactive time) so that it can be used online.

Meeting these objectives is a non-trivial problem. In general, the upper-bound on the number of possible schedules for a TÆMS task structure containing n actions is given in Equation 1. Clearly, for any significant task structure the brute-strength approach of generating all possible schedules is infeasible – offline or online. This expression contains complexity from two main sources. On the "planning" side, the scheduler must consider the (unordered) $O(2^n)$ different alternative different ways to go about achieving the top level

[8] Because there may be alternative ways to perform a given task, and some of the options may not have the same associated deadlines, the scheduler actually balances both meeting hard constraints and the design criteria.

task (for a task structure with n actions). On the "scheduling" side, the scheduler must consider the $m!$ different possible orderings of each alternative, where m is the number of actions in the alternative.

$$\sum_{i=0}^{n} \binom{n}{i} i! \tag{1}$$

In general, the types of constraints present in TÆMS, and the existence of interactions between tasks (and the different *QAFs* that define how to achieve particular tasks), prevent a simple, optimal solution approach. DTC copes with the high-order combinatorics using a battery of techniques. Space precludes detailed discussion of these, however, they are documented in [22]. From a very high level, the scheduler uses:

Criteria-Directed Focusing The client's goal criteria is not simply used to select the "best" schedule for execution, but is also leveraged to focus all processing activities on producing solutions and partial solutions that are most likely to meet the trade-offs and limits/thresholds defined by the criteria.

Approximation Schedule approximations, called *alternatives*, are used to provide an inexpensive, but coarse, overview of the schedule solution space. One alternative models one way in which the agent can achieve the top level task. Alternatives contain a set of unordered actions and an estimation for the quality, cost, and duration characteristics that will result when the actions are sequenced to form a schedule. This, in conjunction with criteria-directed focusing enables DTC to address the "planning" side complexity.

Heuristic Decision Making To address the scheduling side complexity, DTC uses a superset of the techniques used in Design-to-Time [8], namely an iterative, heuristic, process of sequencing out the actions in a given alternative. These action rating heuristics rate each action and the ratings (in DTC) are stratified so that certain heuristics and constraints dominate others. The net effect is a reduction of the $O(n!)$ ($\omega(2^n)$ and $o(n^n)$ by Stirling's Approximation) complexity to polynomial levels in the worst case.

Heuristic Error Correction The use of approximation and heuristic decision making has a price – it is possible to create schedules that are suboptimal, but, repairable. A secondary set of improvement [27, 19] heuristics act as a safety net to catch the errors that are correctable.

The Design-to-Criteria scheduling process falls into the general area of flexible computation [9], but differs from most flexible computation approaches in its use of multiple actions to achieve flexibility (one exception is [10]) in contrast to *anytime algorithms* [4, 18, 25]. We have found the lack of restriction on the properties of primitive actions to be an important feature for application in large numbers of domains. Another major difference is that in DTC we not only propagate uncertainty [26], but we can work to reduce it when important to the client. DTC differs from its predecessor, Design-to-Time[8], in many ways. From a client perspective, however, the main differences are in its use of uncertainty, its ability to retarget processing at any trade-off function, and its ability to cope with both "scheduling" and "planning" side combinatorics.

Design-to-Criteria is not without its limitations; when adapting the DTC technology for use in potentially time critical domains, such as the CEROS anti-submarine warfare information gathering task, shown in Figure 5, we encountered an interesting problem. The satisficing focusing methodology used in Design-to-Criteria leads to poor solutions when combined with hard deadlines and certain classes of very large task structures. Without delving into exhaustive detail, the problem is that in order to cope with the high-order combinatorics in these particular situations, the scheduling algorithm must prune schedule approximations, or alternatives, and develop only a subset of these. Herein lies the problem.

Alternatives are constructed bottom-up from the leaves of the task hierarchy to the top-level task node, i.e., the alternatives of a task are combinations of the alternatives for its sub-tasks. Figure 6 shows the alternative set generation process for a small task structure. Alternatives are generated for the interior tasks T_1 and T_2, and these alternatives are combined to produce the alternative set for the root task, T. The complexity of the alternative generation process is pronounced. A task structure with n actions leads to $O(2^n)$ possible alternatives at the root level. We control this combinatorial complexity by focusing alternative generation and propagation on alternatives that are most likely to result in schedules that "best" satisfice to meet the client's goal criteria; alternatives that are less good at addressing the criteria are pruned from intermediate level alternative sets. For example, a criteria set denoting that certainty about quality is an important issue will result in the pruning of alternatives that have a relatively low degree of quality certainty. After the alternative set for the high-level task is constructed, a subset of the alternatives are selected for scheduling.

For situations in which there is no overall hard deadline, or in which shorter durations are also preferred, the focusing mechanism works as advertised. However, in the CEROS project, we are also interested in meeting real-time deadlines and other hard resource constraints (in contrast to those that are relaxable), and often these preferences are not accompanied by a general preference for low duration or low cost. In these cases, the problem lies in making a *local* decision about which alternatives to propagate (at an interior node) when the decision has implications to the local decisions made at other nodes – the local decision processes are interdependent and they interact over a shared resource, e.g., time or money. Casting the discussion in terms of Figure 6: assume T has an overall deadline of 5 minutes and T_1's alternatives require anywhere from 2 minutes to 20 minutes to complete, and T_2's alternatives are similarly characterized. Assume that quality is highly correlated with duration, thus the more time spent problem solving, the better the result. If the criteria specifies maximum quality within the deadline, the alternatives propagated from T_1 to T will be those that achieve maximum quality (and also have high duration). Likewise with the alternatives propagated from T_2. The re-

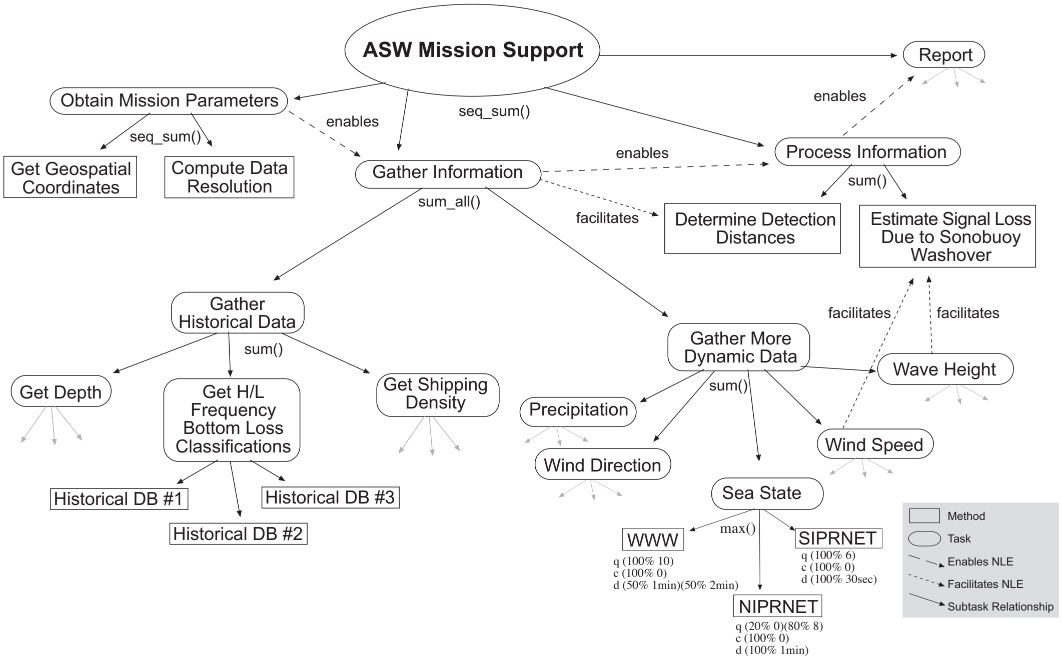

Figure 5: Partial TÆMS Task Structure for Gathering and Processing ASW Mission Information

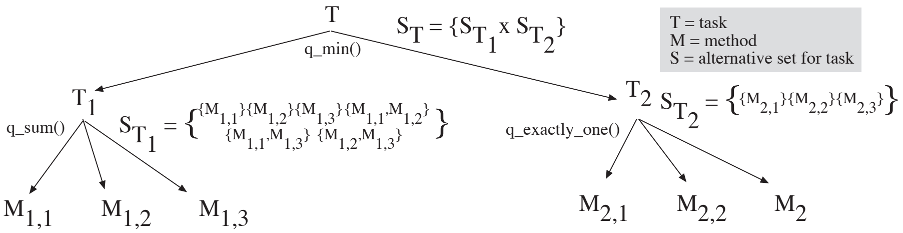
T
$S_T = \{S_{T_1} \times S_{T_2}\}$
q_min()
T = task
M = method
S = alternative set for task
T_1
q_sum()
$S_{T_1} = \left\{ \begin{matrix} \{M_{1,1}\}\{M_{1,2}\}\{M_{1,3}\}\{M_{1,1},M_{1,2}\} \\ \{M_{1,1},M_{1,3}\}\ \{M_{1,2},M_{1,3}\} \end{matrix} \right\}$
T_2
$S_{T_2} = \left\{ \{M_{2,1}\}\{M_{2,2}\}\{M_{2,3}\} \right\}$
q_exactly_one()
$M_{1,1}$
$M_{1,2}$
$M_{1,3}$
$M_{2,1}$
$M_{2,2}$
M_2

Figure 6: Alternative Sets Lead to Cumbersome Combinatorics

sulting set of alternatives, S_T at node T will contain members characterized by high quality, but also high duration, and the scheduler will be unable to construct a schedule that meets the hard deadline. The optimal solution to this problem is computationally infeasible ($\omega(2^n)$ and $o(n^n)$) as it amounts to the general scheduling problem because of task interactions and other constraints.

Two approximate solutions are possible. One approach is to preprocess the task structure, producing small alternative sets at each node that characterize the larger alternative population for that node. Then examining the ranges of alternatives at each node and heuristically deciding on an allocation or apportionment of the overall deadline or cost limitation to each of the interior nodes. This local-view of the overall constraint could then be used to focus alternative production on those that will lead to a root-level set that meets the overall constraint. The other approach, which we have employed, is to detect when the local-view of the decision process is problematic and in those cases sample from the population of alternatives, producing a subset that exhibits similar statistical properties, and propagating these alternatives. This leads to a less-focused set of root level alternatives than the prior approach, but it saves on the added polynomial level expense of the first approach.

Conclusion, Future Work, and Limitations

We have discussed a class of issues in DTC that pertain to modeling and scheduling for hard and soft temporal constraints, resource constraints, and task interactions. Space precludes a full enumeration of the different aspects of DTC that relate to addressing resource limitations – the issue is ubiquitous to the design of the DTC algorithm, the TÆMS modeling framework, and the decisions made by the DTC scheduler. From a very high level, possibly the most important features that relate to resource boundedness is the detailed quantified view of actions, and task interactions, afforded by the TÆMS modeling framework. This, combined with the element of choice present in TÆMS families of plans, sets the foundation for DTC's reasoning about the implications of failures, failing to acquire resources, and violating hard constraints.

In terms of limitations, DTC's approximate solution approach is clearly not optimal in many circumstances. As discussed, this is particularly true when the alternative sets must be severely pruned (focused) to produce solutions. Additionally, in some applications, in which only very specific subsets of the features afforded by TÆMS are employed, custom schedulers may do a better job of balancing the different concerns and finding good solutions. In terms of optimality, it is difficult to compare the performance of DTC to optimal as found via exhaustive generation simply because it is not feasible to generate all possible schedules for realistic task structures. Members of our group are currently working on an MDP-based TÆMS scheduling tool [17, 23] and we plan to measure DTC's performance on smaller applications through this tool.

It is important to note that though DTC takes great pains to produce schedules quickly, the scheduler is not hard real time itself. We cannot make performance guarantees [15] for a given problem instance, though it would be possible to produce such guarantees by classifying similar task structures and measuring scheduling performance offline. At issue is the constraints present in an arbitrary TÆMS task structure. For certain classes of task structures, guarantees without an in-depth preclassification are possible. In practice, the scheduler (implemented in 50,000 lines of C++) is fast and capable of scheduling task structures with 20-40 primitive actions in under 7 seconds on a 600mhz Pentium III machine running Redhat Linux 6.0. A sampling of applications and runtimes are shown in Table 1.

In the table, the first column identifies the problem instance, the second column identifies the number of primitive actions in the task structure, the third column (*UB # R-Alts*) indicates the upper bound on the number of root-level alternatives, the fourth column identifies the upper bound on the number of schedules possible for the task structure ("N/C" indicates that the value is too large for the variable used to compute it). The fifth column (*# Alts R / Total*) identifies the number of alternatives actually produced during the scheduling run – the first number is the number of alternatives produced at the root note and the second number is the total number of alternatives produced during scheduling. The first number is comparable to the upper-bounds expressed in column three. The sixth column shows the number of schedules actually produced. The column labeled *# D Combines* indicates the number of distribution combination operations performed during scheduling – this is particularly informative because nearly all aspects of the scheduling process involve probability distributions rather than expected values. The last three columns pertain to the time spent (in whole seconds) doing different activities, namely producing the set of root-level alternatives, creating schedules from the alternatives, and the total scheduler runtime (which includes some final sorting and other output-related operations). Due in part to the scheduler's use of a particular set of clock functions, which are integer based, there is no variance when the experiments are repeated because the variance pertains to less than whole seconds.

Most of the task structures produced 15 schedules – this is the system default. When fewer schedules are produced it indicates that there are not sufficient alternatives at the root level to produce more schedules. When more than 15 schedules are produced, it indicates that the scheduler's termination criteria was not met – generally caused by a large percentage of zero quality schedules or by there being alternatives that appear better than any schedules generated thus far per the design criteria. The scheduler will work beyond its preset number under these conditions but only to some multiple of the preset. The JIL_translated structure, for example, contains some modeling problems that produce a very large number of zero quality schedules and DTC scheduled up to 4*15 and then halted with a small set of viable schedules.

With respect to scheduler computation overhead and online performance, the time required to schedule these task structures is reasonable given that the grainsize of the structures themselves is much larger than the seconds required to perform the scheduling operation (generally, scheduler overhead is at most 1% of the total runtime of the agent's application). That being said, however, being "appropriately fast" is not necessarily the long term objective and performance guarantees and performance estimates are important research avenues for the future.

One promising area of DTC related research is an offline contingency analysis tool [17, 23] that uses DTC to explore an approximation of the schedule space for a given TÆMS task structure. The use of DTC as an oracle enables the contingency analysis tool to cope with the combinatorics of the general scheduling problem. The contingency analysis methodology determines the criticality of failures within a schedule and for critical points, evaluates the statistical characteristics of the available recovery options. The analysis, while expensive, is appropriate for mission-critical hard deadline situations in which a solution must be guaranteed by a particular time. With DTC's approach, it is possible to start down a solution path (that is appealing even with the possibility of failure) for which there is no mid-stream recovery option that will enable the agent to still produce some result by the hard deadline. DTC will always recover and find whatever options are available, but, because it does not plan for contingency and recovery *a priori*, in hard deadline situations in which solutions must be guaranteed, there is some possibility of unrecoverable failure.

DTC has many different parameter settings not discussed here and it can be made to avoid failure if possible. However, while this covers a certain class of the functionalities offered by the contingency analysis tool, the two are not equivalent. Whereas the best DTC can do is avoid failure if possible, or work to minimize failure, it can only do this for a single line of control. Using the contingency analysis tool, the agent can select a high risk plan of action that also has some potential of a high pay off, but, it can also reason *a priori* about the ability to recover from a failure of the plan. While DTC can be extremely conservative, it cannot plan for both high-risk and recovery concurrently. The choice between DTC and the contingency analysis approach is dependent on the application. For online, responsive control to unplanned events, DTC is most appropriate. For mission-critical situations combined with time for *a priori* offline planning, the contingency approach is most appropriate.

Acknowledgments

We would like to acknowledge the Design-to-Time work of Professor Alan Garvey; said work made this and other developments in TÆMS scheduling possible. We would al-

Task Structure	# Methods	UB # R-Alts	# UB Schedules	# Alts R / Total	# Sched. Prod.	# D Combines	T Prod. Alts	T Prod. Sched.	T Total
Simple Plan Trip	10	1,023	9,864,100	53 / 91	15	8,686	0	1	1
HotWaterHeater1.0	6	63	1,956	45 / 108	19	7,831	0	1	2
DishWasher1.0	14	16,383	N/C	100 / 1018	15	47,136	0	1	2
IHomeRobot	186	2,147,483,647	N/C	50 / 665	15	46,140	0	2	3
Transport_v0	9	511	986,409	48 / 167	15	13,032	0	1	1
BIG_v1.2	26	67,108,863	N/C	50 / 4540	15	273,283	2	3	6
JIL_translated	25	33,554,431	N/C	139 / 3526	60	334,393	2	3	7

Table 1: Scheduler Performance on an Assortment of Different Applications

so like to thank Garvey for his assistance in understanding the details of Design-to-Time when implementing the first stages of Design-to-Criteria. The deadline over-run technique discussed in this paper, where over-run is reflected in a method's quality, is similar to a technique developed and deployed by Garvey in Design-to-Time. The technique was independently developed and deployed in Design-to-Criteria.

Anita Raja also deserves credit for pushing on the issue of recovery in mission critical environments. Her work in contingency analysis via Design-to-Criteria encouraged a clear distinction between situations in which a deeper analysis is worthwhile or required and situations in which online responsiveness is appropriate.

References

[1] Ella M. Atkins, Edmund H. Durfee, and Kang G. Shin. Detecting and Reacting to Unplanned-for World States. In *Proc. of the 14th Natl. Conf. on AI*, July 1997.

[2] Mark Boddy and Thomas Dean. Solving time-dependent planning problems. In *Proc. of the 11th Intl. Joint Conf. on AI*, pages 979–984, Detroit, August 1989.

[3] Ceros-related information gathering project – in progress. Organization url is http://www.ceros.org.

[4] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proc. of the Seventh Natl. Conf. on AI*, pages 49–54, August 1988.

[5] Keith Decker and Jinjiang Li. Coordinated hospital patient scheduling. In *Proc. of the 3rd Intl. Conf. on Multi-Agent Systems*, pages 104–111, 1998.

[6] Keith S. Decker. *Environment Centered Analysis and Design of Coordination Mechanisms*. PhD thesis, University of Massachusetts, 1995.

[7] E. H. Durfee. A cooperative approach to planning for real-time control. In *Proc. of the Wrkshp. on Innovative Approaches to Planning, Scheduling and Control*, pages 277–283, November 1990.

[8] Alan Garvey and Victor Lesser. Design-to-time real-time scheduling. *IEEE Trans. on Systems, Man and Cybernetics*, 23(6):1491–1502, 1993.

[9] Eric Horvitz, Gregory Cooper, and David Heckerman. Reflection and action under scarce resources: Theoretical principles and empirical study. In *Proc. of the 11th Intl. Joint Conf. on AI*, August 1989.

[10] Eric Horvitz and Jed Lengyel. Flexible Rendering of 3D Graphics Under Varying Resources: Issues and Directions. In *Proc. of the AAAI Sympos. on Flexible Computation in Intelligent Systems*, November 1996.

[11] David Jensen et al, Learning Quantitative Knowledge for Multiagent Coordination. *Proc. of AAAI-99*, 1999. Also as UMASS CS Technical Report TR-99-04.

[12] David Jensen et al, Coordinating agent activities in knowledge discovery processes. In *Proc. of Work Activities Coordination and Collaboration Conf.*, 1999. Also available as UMASS Tech Report UM-CS-1998-033.

[13] 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 *Proc. of the 3rd Intl. Conf. on Autonomous Agents*, 1999.

[14] Victor Lesser, Bryan Horling, Frank Klassner, Anita Raja, Thomas Wagner, and Shelley XQ. Zhang. BIG: An agent for resource-bounded information gathering and decision making. *To appear in the AIJ*, 2000.

[15] David J. Musliner, Edmund H. Durfee, and Kang G. Shin. CIRCA: A cooperative intelligent real-time control architecture. *IEEE Trans. on Systems, Man and Cybernetics*, 23(6), 1993.

[16] David J. Musliner, James A. Hendler, Ashok K. Agrawala, Edmund H. Durfee, Jay K. Strosnider, and C. J. Paul. The Challenge of Real-Time AI. *Computer*, 28(1):58–66, January 1995.

[17] 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 Sympos. on Real-Time Systems.

[18] Stuart J. Russell and Shlomo Zilberstein. Composing real-time systems. In *Proc. of the 12th Intl. Joint Conf. on AI*, pages 212–217, Sydney, Australia, August 1991.

[19] Wolfgang Slany. Scheduling as a fuzzy multiple criteria optimization problem. *Fuzzy Sets and Systems*, 78:197–222, March 1996. Issue 2. Special Issue on Fuzzy Multiple Criteria Decision Making; URL: ftp://ftp.dbai.tuwien.ac.at/pub/papers/slany/fss96.ps.gz.

[20] Regis Vincent, Bryan Horling, Thomas Wagner, and Victor Lesser. Survivability simulator for multi-agent adaptive coordination. In *Proc. of the 1st Intl. Conf. on Web-Based Modeling and Simulation*, 1998.

[21] Thomas Wagner, Alan Garvey, and Victor Lesser. Complex Goal Criteria and Its Application in Design-to-Criteria Scheduling. In *Proc. of the 14th Natl. Conf. on AI*, pages 294–301, July 1997. Also as UMASS CS TR-1997-10.

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

[23] Thomas Wagner, Anita Raja, and Victor Lesser. Modeling Uncertainty and its Implications to Design-to-Criteria Scheduling, *Under review Special issue of the AI Journal*, Also as UMASS CS TR-98-51

[24] Ping Xuan and Victor R. Lesser. Incorporating uncertainty in agent commitments. In N.R. Jennings and Y. Lespérance, editors, *Intelligent Agents VI — Proc. of ATAL-99*, Lecture Notes in AI. Springer-Verlag, Berlin, 2000.

[25] S. Zilberstein and S. J. Russell. Optimal composition of real-time systems. *AI*, 82(1):181–214, December 1996.

[26] Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73–83, 1996.

[27] M. Zweben, B. Daun, E. Davis, and M. Deale. Scheduling and rescheduling with iterative repair. In M. Zweben and M. Fox, editors, *Intelligent Scheduling*, Chapter 8. Morgan Kaufmann, 1994.