Complex Goal Criteria and Its Application in Design-to-Criteria Scheduling *†

Thomas Wagner
Computer Science Department
University of Massachusetts
Amherst, MA 01003
Email: wagner@cs.umass.edu

Alan Garvey
Department of Computer Science
Truman State University
Kirksville, MO 63501
Email: garvey@cs.umass.edu

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

Abstract

Difficult real-time AI problems require a means for expressing multi-dimensional and dynamic goal criteria and a principled model for satisficing to best meet the criteria. In the context of the Design-to-Criteria task scheduling paradigm, we define a new general client specification metaphor for describing such complex goal criteria or utility attributes, and couple it with a principled evaluation model for using the criteria. The criteria specification and corresponding evaluation mechanism are used throughout the Design-to-Criteria scheduling process to focus scheduling activities on solutions and partial solutions that are most likely to meet the criteria, i.e., to result in the focused production of custom satisficing schedules. Examples of the power of the approach at reducing the complexity of the scheduling task and designing custom satisficing schedules, are shown.

Introduction

We have developed a new domain independent flexible computation (Zilberstein 1996; Horvitz, Cooper, & Heckerman 1989; Horvitz & Lengyel 1996) approach to task scheduling, based on the Design-to-Time (Garvey & Lesser 1993) work in real-time AI, called *Design-to-Criteria* (Wagner, Garvey, & Lesser 1997b; 1996; 1997a). The three distinguishing features of Design-to-Criteria are the ability to reason about the utility attribute trade-offs of different solutions based on different goal criteria, the ability use these utility attribute trade-offs to focus every step of the scheduling process, and the ability to do these activities from a satisficing perspective. Satisficing with respect to the scheduling process itself enables the scheduler to produce results when computational combinatorics prevent an optimal solution. Satisficing with respect to meeting the criteria enables the scheduler to produce a result that adheres to the spirit of the goal when the criteria cannot be satisfied perfectly due to environmental and resource constraints. We use the term *evaluation* to describe this reasoning process, where one possible solution or partial solution is considered against its peers and satisficed against a set of desired goal criteria that define a utility function. Ubiquitous satisficing in the scheduler is based on a new, inexpensive to compute, general approach to satisficing evaluation coupled with a new domain independent client criteria specification metaphor.

To ground further discussion consider the sample information gathering task structure, written in the TÆMS (Decker & Lesser 1993) language, shown in Figure 1. The task structure models multiple different approaches for gathering information about WordPerfect via the WWW. A set of satisficing schedules produced by the Design-to-Criteria scheduler using the evaluation mechanism and four different sets of evaluation criteria, is shown in Figure 2. Schedule A is constructed for a client interested in a fast, free, solution with any non-zero quality. Schedule B suits a client who wants a timely and free solution, but wants less uncertainty about the expected quality of the results. Schedule C is constructed for a user interested in a good quality, free, solution that can be obtained while she goes for a cup of coffee. Schedule D is generated to meet the criteria of a fourth individual who is willing to pay and wait for a high-quality response.

This very simple example provides a good illustration for the reasoning behind the new evaluation approach. *The goodness of a particular solution is entirely dependent on the client's complex objectives.* To

† This material is based upon work supported by the National Science Foundation under Grant No. IRI-9523419, the Department of the Navy, Office of the Chief of Naval Research, under Grant No. N00014-95-1-1198, and via a subcontract from Boeing Helicopter which is being supported by DARPA under the RaDEO program (contract number 70NANB6H0074). The content of the information does not necessarily reflect the position or the policy of the Government, National Science Foundation, or Boeing Helicopter and no official endorsement should be inferred.

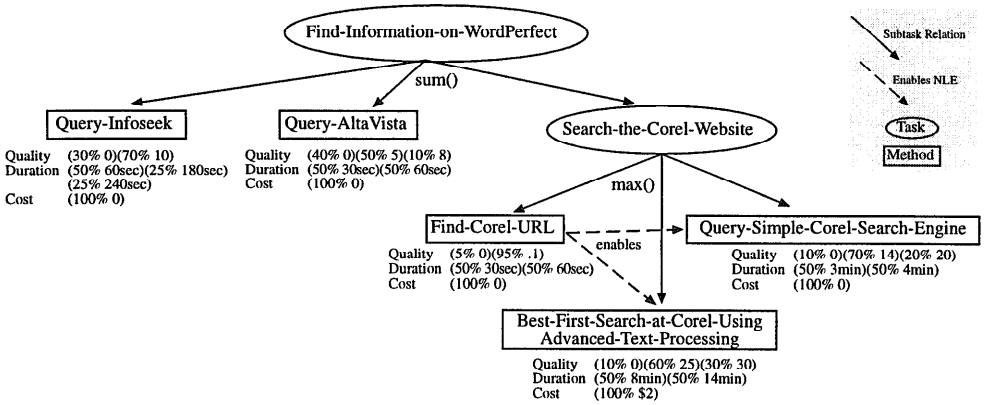


Figure 1: Information Gathering Task Structure

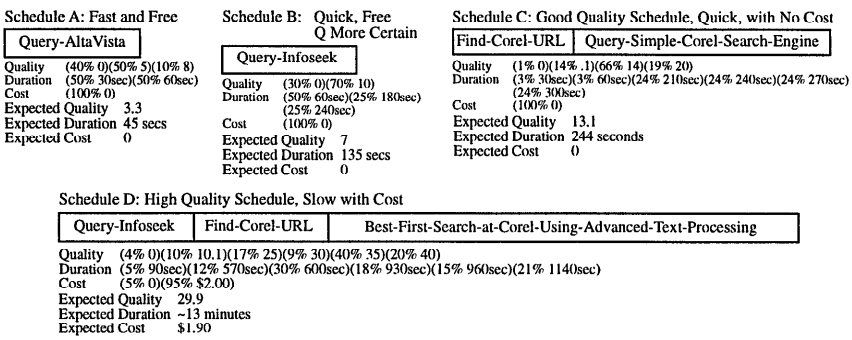


Figure 2: Four Satisficing Schedules

support integration with humans and other complex components, in different applications and hard environments, flexible computation-based AI problem solving systems must support dynamic complex goal criteria and the criteria must be used to decide what goals to achieve, what actions to use to accomplish the goals, the resources to assign, and the scheduling of the actions. The rationale is:

- The goal criteria must be dynamic for two reasons. First, because different clients have different priorities. Second, the ability of the scheduler/problem solver to produce a good solution is dependent on the options available. If the scheduler cannot produce an acceptable satisficing solution for the task structure in question the client may want to submit new criteria and try again. The criteria delineates a view into a pool of possible solutions and it is possible that no solutions actually reside within the view. We discuss interactive refinement of criteria later in this paper.
- Most work in real-time AI emphasizes producing high quality solutions within a given time allotment. But the problems tackled by complex real-time AI applications are themselves complex and solutions are often characterized in many dimensions, not just quality and duration. For example, in the information gathering domain cost and uncertainty are necessary attributes in addition to the standard quality and duration. Complex problems dictate that the evaluation criteria be correspondingly complex and richly expressive. While expressive, the criteria must also be understandable to the client and evaluation based on the criteria must be consistent with the client's semantic model of the criteria so the results suit the client's needs.
- As most AI problem solvers work in a world of explosive combinatorics, the entire problem solving system must work to meet the specified criteria at every step. Work cannot be wasted constructing solutions or partial solutions that do not address the client's objectives. In Design-to-Criteria scheduling the solution space is typically exponential in the number of primitive actions and we control the $O(2^n)$ combinatorics using the new evaluation mechanism. Processing at all points in Design-to-Criteria, from the generation of partial approximate solutions to the creation of schedules, is targeted at the client's criteria.

The new evaluation model operates to determine a principled measurement of overall utility based on

multiple attributes or evaluation dimensions and a dynamic set of goal criteria. The tenants of *relativity* and *proportionality* are the keystones of the model. Relativity is important because in an imperfect sub-optimal situation, the goodness of one solution is *relative* to the other possible solutions, i.e., the objective is to make satisficing choices. Proportionality is a major concern because the weight of a particular attribute must be proportional to the weight or importance specified by the goal criteria – differences of scale must not affect the evaluation mechanism.

The new client specification metaphor, *importance sliders*, also adheres to the relativity and proportionality foundation. The slider metaphor enables clients, users or other systems, to define the relative importance of multiple attributes with respect to four classes of concerns: raw goodness, thresholds and limits, certainty, and thresholds on certainty. We preface further detail on the new evaluation functions and specification metaphor with background information on the TÆMS modeling framework and the Design-to-Criteria scheduling task.

TÆMS and Design-to-Criteria Scheduling

TÆMS (Task Analysis, Environment Modeling, and Simulation) is a domain independent task modeling framework used to describe and reason about complex problem solving processes. TÆMS models serve as input to the Design-to-Criteria scheduler. 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 intimate details of each primitive action. All primitive actions in TÆMS, called *methods*, are statistically characterized in three dimensions: quality, cost and duration. Quality is a deliberately abstract domain-independent concept that describes the contribution of a particular action toward achieving the overall goal and the relative importance of its contribution. Thus, different applications have different notions of what corresponds to model 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. The statistical characteristics of the three dimensions are described via discrete probability distributions associated with each method (see Figure 1).

Design-to-Criteria scheduling is the process of custom tailoring a way to achieve the high-level task via the actions described in the model to fit a particular client's quality, cost, duration, and certainty criteria or needs. Thus satisficing evaluation in the Design-to-Criteria context means satisficing in three attribute dimensions: quality, cost, and duration, and three uncertainty dimensions: uncertainty about quality, uncertainty about cost, and uncertainty about duration. In applications where uncertainty is not represented or appropriate, the bulk of the approach that follows will still work as advertised. Readers should also note that though the remainder of the paper discusses satisficing in these dimensions the evaluation approach is extensible to n dimensions. For example, in a perceptual interpretation task, the quality attributes could be more detailed and broken down into three independent attributes: solution precision, solution certainty, and solution completeness.

Satisficing evaluation is ubiquitous in the new Design-to-Criteria paradigm. Prior to actually building schedules the scheduler must enumerate the different ways that the high-level tasks can be achieved. Each "way" is a cheap to compute schedule approximation called an *alternative*; associated with each alternative is an unordered set of primitive actions and an estimate for the quality, cost, and duration distributions that will result from scheduling the alternative. Alternatives differ from schedules in that the ordering for the primitive actions has not yet been defined and the attribute estimates are computed without regard for complex task interactions. 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 subtasks. The problem is that a TÆMS task structure with n methods leads to $O(2^n)$ possible alternatives at the root level. Satisficing evaluation is used to control this combinatorial complexity – intermediate alternative sets are pruned using the evaluation mechanism and the client's criteria. This enables the scheduler to avoid doing work propagating alternatives that are unlikely to lead to schedules that meet the client's specified criteria.

After the alternative set for the high-level task is constructed, a subset of the alternatives are selected for scheduling. (The complexity of turning alternatives into schedules, $O(m!)$ for exhaustive search and low-order polynomial for the heuristic approach we use, prevents scheduling all alternatives.) Satisficing evaluation is used at this stage to select the alternatives that are most likely to lead to schedules that fit the client's criteria. We will empirically demonstrate the efficacy of this later in the document.

As each schedule is created, experts critique the schedule and suggest improvements by adding new

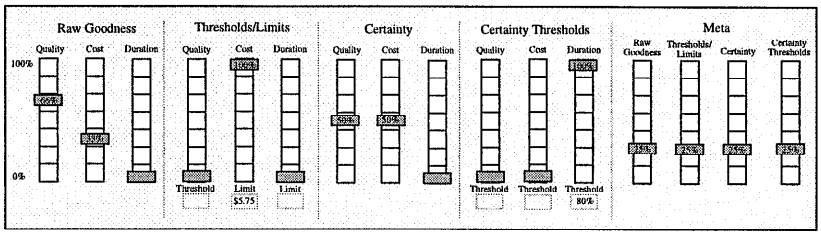


Figure 3: A Slider Set Describing Particular Criteria

alternatives to the alternative set for the root task. Again, satisficing evaluation is used to re-rank the candidate alternatives according to the client's criteria. After the schedule building process terminates (for a variety of conditions) the set of candidate schedules is ranked using the evaluation mechanism and the schedule that best meets the client's criteria is selected for execution. Design-to-Criteria's constant use of satisficing evaluation with dynamic evaluation criteria makes the scheduler fully targetable for any application and suitable for even dynamic settings where negotiation between the scheduler client and the scheduler is used to iteratively refine the scheduling process.

Sliders - The Client Criteria Specification Metaphor

The objective of the evaluation approach is to translate a client's needs, expressed as evaluation criteria, into choosing the course of action that best meets the criteria. Clients are good at expressing and reasoning about the *relative* importance of quality, cost, and duration, but they are less good at assigning particular values that denote goodness. Thus, our evaluation approach operates on the conceptual notion of *importance sliders* that clients "set" for each dimension in the criteria set. The importance sliders, which take on percentage values from 0 to 100, describe the relative importance of each of dimension in a domain independent fashion. Using the sliders, client applications or users can express the notions like "quality is twice as important as cost and duration is half as important" or "quality and duration are equally important but cost is no issue."

While we have introduced sliders in a general sense there are actually five sets of sliders used in the criteria specification process, some of which are accompanied by absolute requirements in the form of thresholds or limits. We should note that the sliders take on percentage values, with the constraint that each bank's sliders sum to 100%, purely for semantic reasons. The entire evaluation approach will work with any range of values and without the 100% sum constraint. The slider sets, shown in Figure 3, are:

Raw Goodness This slider set contains sliders for each dimension, quality, cost, and duration. Its purpose is to describe the relative importance of each dimension. For example, setting quality to 50% and cost and duration to 25% expresses the notion that quality is twice as important as each of the other dimensions and that it should weigh twice as heavily as each when evaluating schedules or alternatives.

Threshold and Limits This slider set also contains sliders for each dimension, however, in this set each slider is paired with a value, or a gradual utility function, that denotes the minimum desired threshold for the quality dimension or the maximum limit for cost or duration. The separation of limits and thresholds from overall goodness allows clients to specify concepts like "Cost, quality and duration are equally important in general, but schedules whose cost is under my limit are particularly valuable."

Where utility functions are used instead of values to delineate changes in utility, the functions describe the range in which utility begins to increase as quality increases, or the range where utility decreases as cost or duration increase. The utility function specification lets clients express notions like "Overall quality and cost are equally important and I want a solution in five minutes, but I'll grudgingly wait ten minutes for a high-quality solution."

Note that the limits and thresholds describe quantities that schedules or alternatives must exceed in order to get points from this set of sliders, i.e., schedules that fail to beat thresholds and limits may still be returned for execution if they best satisfice over the criteria set as a whole. The issue of satisficing with respect to hard constraints is beyond the scope of this paper but the solution lies in negotiation.

Certainty This slider set defines how important reducing uncertainty in each dimension is relative to the other dimensions. In particular applications it may be more desirable to pick a slower, more costly schedule that returns lower expected quality because the certainty about the resulting quality is very high.

Certainty Thresholds This bank is analogous to the thresholds/limits bank above except that this bank focuses on the uncertainty associated with

quality, cost, and duration. Schedules or alternatives whose certainty in a particular dimension meet or exceed the defined threshold are preferred. This enables clients to expression notions like "certainty in the quality dimension is not important as long as the schedule is at least 80% likely to produce the expected quality value or one better," as opposed to raw certainty objectives like "certainty in the quality dimension is important." As with the thresholds/limits sliders, the thresholds can be gradual, rather than a single value, and specified via a function or curve.

Meta This slider set relates the importance of the four previous slider sets. This separation allows clients to focus on relating quality, cost and duration with each other in each of the cases above, then to "step back" and decide how important each of the different aspects are relative to the others.

In the example slider set, shown in Figure 3, quality is the most important general factor with cost being one half as important and duration being not important at all. In terms of thresholds, quality and duration have none, but schedules whose cost is below $5.75 are preferred. Schedules whose expected quality and cost values are more certain are also preferred and uncertainty about duration is not an issue as long as schedules meet or exceed the 80% duration certainty threshold. Relating the four sets of criteria together, they are all equally (25%) important and thus all contribute equally to the overall ranking. Mapping this example to the real world, this could describe the criteria of an individual performing research on the web who does not need the information in a timely fashion, has limited funds in his or her pocket, wants good quality information, but also wants to be certain of the cost and quality of the proposed solution before committing to a course of action, and is scheduling activities later in the day based on the expected search duration.

Mapping Sliders to Ratings

After defining the slider sets, the problem then becomes how to use the criteria to produce results that match expectations. When determining schedule or alternative "goodness," alternatives or schedules are rated using the relative importances expressed on the sliders. We associate a rating component with each of the slider banks, excluding the meta bank, and then combine them according to the relative weights expressed in the meta slider bank. The omnipresent themes in the rating calculations are relativity and proportionality.

In general, we calculate the rating component for a given slider bank by calculating sub-components for each dimension: quality, cost, and duration. Each dimension's sub-component is computed by looping over the set of items to be evaluated and normalizing each item's expected value or expected probability (in the uncertainty case) for that particular dimension, and then multiplying the result by the relative importance as expressed in the slider. It is crucial to normalize the values to a common scale so that in domains where one dimension, say quality, is exponentially larger than the others it does not dominate the ratings disproportionately. Scaling based on the observed minimum and maximum values for a given dimension is similarly important. With the exception of the threshold/limit case we are interested in relative goodness between alternatives or schedules. By using minima and maxima that are derived from the set of items being rated, we automatically scale the grain size to define relative differences in the items. For example, say Schedule A has expected quality of 4.7, Schedule B has expected quality of 4.2, and Schedule C has expected quality of 4.0. In absolute numerical terms Schedule A is "a little" better than both B and C. However, in relative terms, Schedule A is by far the best of the possible schedules. The notion of relative scaling will become more clear from the equations that follow.

We calculate the rating component for the first slider bank, that describes the raw goodness of a particular dimension, as follows:

1. Find the min and max expected values for quality, cost, and duration that occur in the set of schedules or alternatives being rated.
2. Loop over the set of alternatives or schedules to be rated and calculate the raw goodness rating for each by calculating the quality, cost, and duration subcomponents as follows in Steps 3 and 4.
3. Let *this.eq* denote the expected quality value of the alternative or schedule under consideration. Its quality sub-component is a function of the the percentage of quality achieved by *this.eq* relative to the min and max, min_q and max_q, quality values of the set of items being rated, scaled by the raw goodness quality slider, RG_slider_q and the total number of points in the raw goodness bank.

$$rating_q = \frac{(this.eq - min_q)}{max_q - min_q} * \frac{RG\_slider_q}{\sum_{i=q}^{d,c} RG\_slider_i}$$

4. Duration is different than quality in that more duration is a less good thing. Whereas with the quality related equation, achieving the best quality of all items in the set should bring the highest reward, in this case, achieving the least duration of all items in

the set should bring the highest reward. Cost is like duration in that lower cost is better.

$$rating_d = \frac{(max_d - this.ed)}{max_d - min_d} * \frac{RG\_slider_d}{\sum_{i=q}^{d,c} RG\_slider_i}$$

$$rating_c = \frac{(max_c - this.ec)}{max_c - min_c} * \frac{RG\_slider_c}{\sum_{i=q}^{d,c} RG\_slider_i}$$

5. The quality, duration, and cost sub-components are then summed to obtain the aggregate raw goodness rating component.

The threshold or limit rating component is likewise composed of three sub-components. Originally, we modified the equations above by replacing the derived quality minimum, and the derived cost and duration maximums, with the client provided threshold/limits. However, this approach leads to rewards for high relative quality, and low relative cost and duration, from both the threshold/limit bank and from the raw goodness bank. The resulting ratings often didn't map well to the semantic model presented by the sliders. Thus, the current threshold/limit rating components are even more simple to compute – quality above the specified threshold, and cost and duration below the specified limits, are rewarded according to the relative settings of the quality, cost, and duration sliders. Beating a threshold or a limit is rewarded the same regardless of how well a particular schedule or alternative beats the threshold or limit. If a gradual utility function is used in lieu of a single threshold/limit value, the reward is scaled by the utility percent associated with the particular value.

The certainty rating component is different from the previous components because it does not look at quality, cost, and duration values, but at the uncertainty associated with these values. Consider the quality case. The general idea is to reward alternatives or schedules based on how likely it is that a quality value that meets or exceeds the expected value will actually occur. (An alternate interpretation is to determine the probability that the actual value will fall near the expected value, on the upside or the downside.) Thus we compute the probability that the quality, as expressed by the discrete probability distribution, is greater than or equal to the expected value, we then normalize and scale the probability as with the previous components, and finally multiply by the proportion of points allocated to the certainty quality slider. Consider a partial example, if an alternative has a simple quality distribution that denotes 25% of the time 0 quality will result and 75% of the time quality 10 will result, its resulting expected quality value is 7.5. Contrast this with an alternative whose quality distribution denotes that 50% of the time 0 quality will result and 50% of the time 15 quality will result; its expected quality is also 7.5. However, the probability that the first alternative will generate a quality value greater than or equal to the expected value is .75 whereas the second alternative's probability is only .50. This is the gist of the certainty rating sub-components – the more certain that the expected value, or a better value, will occur, the greater the reward. The calculation procedure is similar to the raw goodness procedure, though the focus is always on probabilities and probabilities of the items being rated are normalized using the derived min and max probabilities for the set. For example, to compute the quality rating subcomponent:

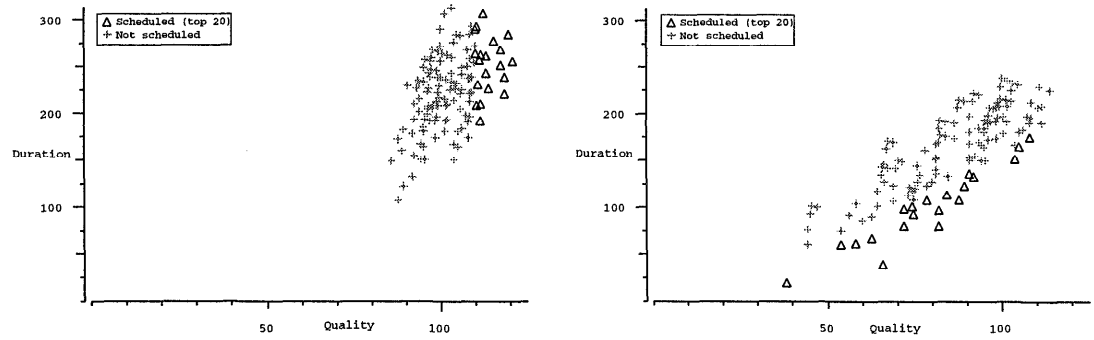
$$r_q = \frac{(Prob(this.q >= this.eq) - min\_probability_q)}{max\_probability_q - min\_probability_q} * \frac{Certainty\_slider_q}{\sum_{i=q}^{d,c} Certainty\_slider_i}$$

The certainty threshold rating component is computed similarly to that of the thresholds/limits rating component, with the exception that the certainty about the given dimension is the theme rather than the expected value of the given dimension. In other words, items whose certainty in a particular dimension meet or exceed the specified threshold are preferred. As above, certainty about quality is the probability that the actual resulting quality value will be greater than or equal to the expected quality value; certainty about cost is the probability that the resulting cost will be less than or equal to the expected cost. Certainty about duration is computed in a similar fashion. As with the thresholds/limits component, if the threshold is defined via a gradual utility function the reward for a particular component is scaled by the utility percent associated with that particular certainty value.

After computing the raw goodness, threshold/limit, certainty, and certainty threshold rating components, the alternate or schedule rating is computed by weighting the rating components according to the relations specified by the meta sliders.

$$\begin{aligned} overall\_rating &= rating_{RG} * meta\_slider_{RG} \\ &+ rating_{TL} * meta\_slider_{TL} \\ &+ rating_{C} * meta\_slider_{C} \\ &+ rating_{CT} * meta\_slider_{CT} \end{aligned}$$

As the evaluation mechanism is used to reduce the computational work required by the scheduling process, it is important to realize that the above components are inexpensive to compute. The process of finding the minima and maxima and calculating the sub



(a) Alternatives for Client A

(b) Alternatives for Client B

Figure 4: Alternatives for the Clients

components is $O(n)$, where n is the number of items being rated. Additionally, the constants involved are very small as most of the values used in the computations are computed elsewhere and stored. Even the cost of sorting the items after they are rated, $O(nlogn)$, or selecting which m items to keep from the set, $O(m * n)$ where $m < n$, is easily dominated by other factors in the scheduling process.

Empirical Example

To see the evaluation approach in action consider a simple empirical Design-to-Criteria illustration. Two different clients are interested in schedules for a moderately complex task structure that has 815 alternative ways and approximately $2 * 10^{13}$ possible schedules (in the unprunned case) to accomplish the task. Client A is interested only in raw quality, thus the raw quality slider is set to 100% and the meta slider for raw goodness is also set to 100%. Client B is interested in raw quality and raw duration equally, thus the raw quality and duration sliders are set to 50% each and the meta slider for raw goodness is set to 100%.

The quality and duration attributes of the alternative set generated for each client is shown in Figures 4(a) and 4(b). Note that in Figure 4(a) the generated alternatives are generally of higher quality and higher duration than those generated in Figure 4(b). Note also that the alternatives scheduled in Figure 4(a) are those whose quality estimates are the highest, while in Figure 4(b) the alternatives that trade-off high-quality for short-duration are selected for scheduling.

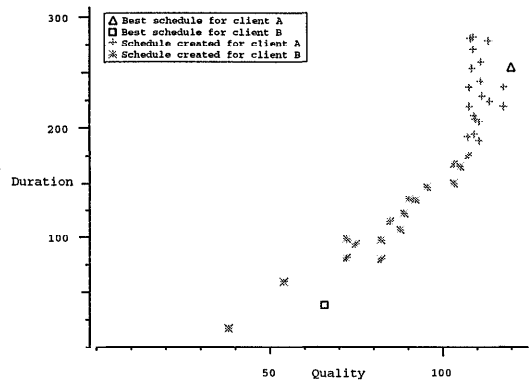
Figure 5(a) shows the schedules generated from the selected alternatives for the criteria sets. For client A, where quality is the only issue, the schedule selected for execution is the one that has the highest expected quality. For client B, where quality and duration are equally important, the best schedule is not one that returns the greatest quality for the least duration, but is instead one that achieves the best quality/duration trade-off – as quality and duration are equally important, this is rational.

Figure 5(b) shows the combined results of executing the selected schedules for the two clients in an unbiased execution environment, where the distributions seen by the scheduler are good models of what actually occurs. In keeping with the estimates of the alternatives and the refined expectations of the schedules, client A's runs returned better quality and client B's runs returned lower quality but also lower duration. The stacking phenomena in B's executions is due to the low number of actions in B's fast schedules; there are fewer possible quality values for B's execution in this unbiased environment.

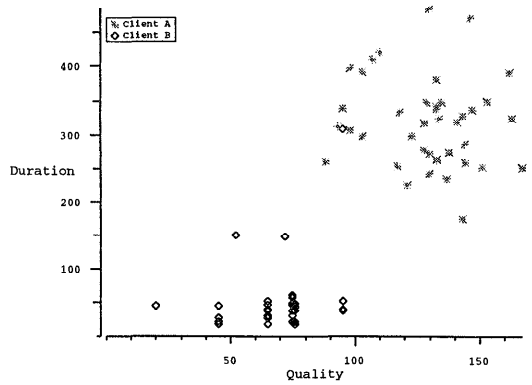
Conclusion and Future Work

Satisficing evaluation is used throughout the new, flexible computation oriented, Design-to-Criteria scheduling paradigm to target all problem solving toward the client's goal criteria. The scheduler's ability to satisfice is the result of a new domain independent evaluation approach coupled with a new client criteria specification metaphor, sliders. The evaluation approach is based on a foundation of relativity and proportionality so that the complex dynamic goal criteria generate desired, understandable, and rational results. The evaluation approach and the criteria specification metaphor are general and can be applied to other applications and domains.

Future work in Design-to-Criteria revolves around refining the interface between the scheduler and other



(a) Schedules for Clients A and B



(b) Execution Results for Clients A and B

Figure 5: Schedules and Results

complex AI components and/or humans. Interactive negotiation (Garvey, Decker, & Lesser 1994) between the client and the scheduler could control and refine satisficing activities as they happen. With the current model of criteria specification followed by application, it is possible that none of the generated schedules satisfactorily meet the client's ideal needs (though the one that best satisfices to meet the criteria will be returned). In this case, the client may prefer an alternate set of criteria rather than taking a satisficing view of the original criteria. Interactive negotiation during the alternative generation and evaluation phases could refine client expectations based on the *estimates* associated with the alternatives. This would enable the scheduler to adjust its intermediate processing to align with the client's refined criteria before any work is spent building schedules. Negotiation during the scheduling phase could help refine the criteria based on the features of schedules as they are produced. The refined criteria would then alter the selection of alternatives and retarget the scheduling activity. Negotiation during the scheduling process is clearly the next step in exploiting and leveraging the power of the Design-to-Criteria paradigm.

References

Decker, K. S., and Lesser, V. R. 1993. Quantitative modeling of complex environments. *International Journal of Intelligent Systems in Accounting, Finance, and Management* 2(4):215–234. Special issue on "Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior".

Garvey, A., and Lesser, V. 1993. Design-to-time real-time scheduling. *IEEE Transactions on Systems, Man and Cybernetics* 23(6):1491–1502.

Garvey, A.; Decker, K.; and Lesser, V. 1994. A negotiation-based interface between a real-time scheduler and a decision-maker. In *AAAI Workshop on Models of Conflict Management.*

Horvitz, E., and Lengyel, J. 1996. Flexible Rendering of 3D Graphics Under Varying Resources: Issues and Directions. In *Proceedings of the AAAI Symposium on Flexible Computation in Intelligent Systems.*

Horvitz, E.; Cooper, G.; and Heckerman, D. 1989. Reflection and action under scarce resources: Theoretical principles and empirical study. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.*

Wagner, T.; Garvey, A.; and Lesser, V. 1996. Satisficing Evaluation Functions: The Heart of the New Design-to-Criteria Paradigm. UMASS Department of Computer Science Technical Report TR-96-82.

Wagner, T.; Garvey, A.; and Lesser, V. 1997a. Criteria-Directed Heuristic Task Scheduling. UMASS Department of Computer Science Technical Report TR-97-16.

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

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