Design-to-Criteria Scheduling: Managing Complexity through Goal-Directed Satisficing [*†]

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

Alan Garvey
Div. of Math & 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

Scheduling complex problem solving tasks, where tasks are interrelated and there are multiple different ways to go about achieving a particular task, is an imprecise science and the justification for this lies soundly in the combinatorics of the scheduling problem. Intractable problems require approximate solutions. We have developed a new domain-independent approach to task scheduling called *Design-to-Criteria* that controls the combinatorics via a satisficing methodology and custom designs schedules to meet a particular client's goal criteria. In Design-to-Criteria, criteria-directed focusing, approximation, and heuristics, in conjunction with soft client goal criteria are used to make the scheduling problem tractable.

Introduction

With the advent of open computing environments adaptability in software applications is critical. Since open environments are less predictable, applications must be able to adapt their processing to the available resources and the different goal criteria set by different clients. An interesting and difficult scheduling problem arises in adaptive systems when there are multiple different ways to achieve tasks and the tasks are interdependent, i.e., the result of one subtask affects the performance, characteristics, or outcome of another subtask in quantifiable ways. The combinatorics possible from even simple interrelated tasks of this type are significant and the problem of scheduling a sequence of actions, given complex goal criteria and limited time, is intractable.

[*]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.

[†] A version of this paper appears in the Working Notes of the AAAI-97 Workshop on Building Resource-Bounded Reasoning Systems, July 1997.

We have developed a new domain independent flexible computation approach to task scheduling called *Design-to-Criteria*. The most distinguishing features of Design-to-Criteria scheduling are the ability to reason about the utility attribute trade-offs of different solutions based on different goal criteria, the ability to 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 goal 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.

Our research focuses on a class of computational task structures where there are typically multiple different actions for performing a particular task, each action has different statistical performance characteristics, and uncertainty about the outcomes of actions is ubiquitous. The input to the scheduling system is a model describing such task structures in the TÆMS (Decker 1995b) domain-independent hierarchical task modeling framework. In TÆMS primitive actions, called *methods*, are modeled statistically 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 the overall problem solving objective. Duration describes the amount of time that a method will take to execute. Cost describes the financial or opportunity cost inherent in performing the action modeled by the method. As with most hierarchical representations the high-level task is achieved by achieving some combination of its subtasks. Accordingly, since different methods have different quality, cost, duration, and certainty trade-offs, different solutions and partial solutions also have different characteristics and different trade-offs. Hard and soft interactions between tasks, called *NLEs* (non-local effects), are also represented in TÆMS and the effects of the interactions are reasoned about statistically during scheduling.

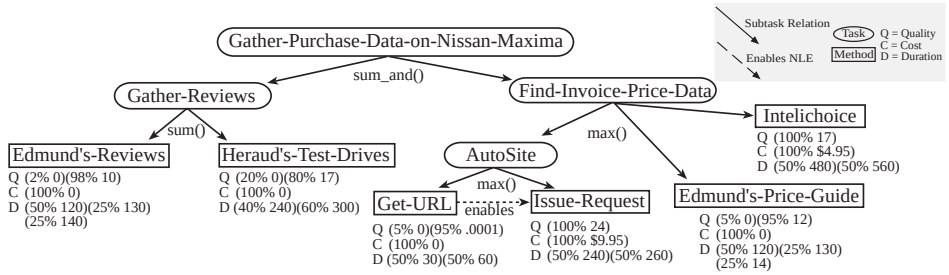

Figure 1: TÆMS Task Structure for Gathering Auto Purchase Information

Schedule A: Fast and Free
Edmund's-Reviews | Edmund's-Price-Guide
Q (~0% 0)(5% 10)(2% 12)(93% 22)
C (100% 0)
D (25% 240)(25% 250)(31% 260)(12% 270)(6% 280)
Expected Q: 21 Q Certainty: 93%
Expected C: 0 C Certainty: 100%
Expected D: 255 seconds D Certainty: 50%

Schedule B: High Quality Certainty, Moderate Cost
Edmund's-Reviews | Intelichoice
Q (2% 17)(98% 27)
C (100% $4.95)
D (25% 600)(12% 620)(31% 680)(19% 700)
Expected Q: 26 Q Certainty: 98%
Expected C: $4.95 C Certainty: 100%
Expected D: 647 seconds D Certainty: 50%

Schedule C: Good Quality, Moderate Cost, Slow
Edmund's-Reviews | Heraud's-Test-Drives | Intelichoice
Q (~0% 17)(20% 27)(2% 34)(78% 44)
C (100% $4.95)
D (20% 840)(19% 900)(31% 920)(19% 980)(11% 1000)
Expected Q: 40 Q Certainty: 78%
Expected C: $4.95 C Certainty: 100%
Expected D: 920 seconds D Certainty: 70%

Schedule D: High Quality, High Cost, Moderate Duration
Edmund's-Reviews | Heraud's-Test-Drives | Get-AutoSite-URL | Issue-AutoSite-Request
Q (1% 0)(4% 27)(19% 34)(2% 41)(74% 51)
C (100% $9.95)
D (20% 630)(31% 690)(24% 720)(19% 740)(6% 760)
Expected Q: 46 Q Certainty: 74%
Expected C: $9.95 C Certainty: 100%
Expected D: 698 seconds D Certainty: 51%

Figure 2: Four Satisficing Schedules

A simplified example of a TÆMS task structure for gathering auto purchase information via the Web is shown in Figure 1. The oval nodes are tasks and the square nodes are methods. The top-level task is to *Gather-Purchase-Data-on-Nissan-Maxima* and it has two subtasks *Gather-Reviews* and *Find-Invoice-Price-Data.* The top-level task accumulates quality according to the *sum_all() quality accumulation function* (qaf)[1] so both of its subtasks must be performed to satisfy the objective. The *Gather-Reviews* task has two methods, query *Edmund's-Reviews* and query *Heraud's-Test-Drives.* These methods are governed by a *sum()* qaf thus the power-set of the methods minus the empty set may be performed to achieve the tasks, i.e., *Edmund's* may be queried, *Heraud's* may be queried, or both may be queried. The *Find-Invoice-Price-Data* task has three subtasks, two of type method and one of type task, governed by the *max()* qaf which is analogous to an OR relationship. Note the decomposition of the obtain invoice via *AutoSite* task into two methods, one that locates the URL and one that issues the query. The *enables* NLE between the URL finding method and the query method, in conjunction with the low quality associated with the URL finding method, indicate that finding the URL is necessary for task achievement but that it contributes very little to achieving the task relative to the method that actually obtains the pricing report.

The central objective in Design-to-Criteria scheduling is to cope with the combinatorial explosion of possibilities while reasoning about a particular set of client goal criteria and the trade-offs presented by different solutions and partial solutions. In other words, scheduler client applications or users specify the design criteria and the scheduler *designs* a schedule to best meet the criteria, if possible given the task model. A set of satisficing schedules produced by the Design-to-Criteria scheduler, for the sample task structure, using four different sets of 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 conservative client who is interested primarily in certainty about quality achievement. Schedule C is designed for a client who wants high quality information, is willing to wait a long time for an answer, and is only willing to spend $5 on the search. Schedule D is meets the criteria of a client who wants the highest possible quality, is willing to spend $10, and wants the gathered data in 15 minutes or less.

This work falls into the general area of flexible computation (Horvitz, Cooper, & Heckerman 1989), but differs from most flexible computation approaches in its use of multiple actions for task achievement (one exception is (Horvitz & Lengyel 1996)), in its first class treatment of uncertainty, and in its ability to use uncertainty information in the selection of methods for execution. Much work in flexible computation makes use of *anytime algorithms* (Dean & Boddy 1988; Russell & Zilberstein 1991; Zilberstein & Russell 1995), algorithms that always have an answer at hand and produce higher quality results as they are given more time, up to a threshold. Our multiple

[1] Qafs define how a given task is achieved through its subtasks or methods. The *sum_all()* qaf means that all of the subtasks must be performed and that the task's quality is a sum of the qualities achieved by its subtasks.

methods approach can model any activity, including anytime algorithms, that can be characterized statistically and we place no constraints on the statistical behavior of the activities in question. In our work, uncertainty is a first class concept that both appears in the statistical descriptions of the available methods and is propagated and related as schedules and schedule approximations are generated. Unlike most work in anytime algorithms that focuses on the propagation of uncertainty (Zilberstein 1996), we can also include uncertainty and uncertainty reduction in the goal criteria and focus work on reducing uncertainty when important to the client (Wagner, Garvey, & Lesser 1997b). This ability stems from our task model's representation of alternative ways to perform various tasks. Because multiple methods often exist to perform tasks, we can reason about the quality, cost, duration, and uncertainty trade-offs of different actions when determining which actions to perform, achieving the best possible overall results.

Design-to-Criteria Scheduling

The fundamental premise of our work is that *the goodness of a particular solution is entirely dependent on a particular client's complex objectives* and that *different client's have varying objectives.* Thus the scheduling process must not only consider the attribute trade-offs of different solutions, but must also do so dynamically. Furthermore, the scheduling process must be efficient as the application domains we study typically involve agents acting in the world in real-time. Because of the inherent uncertainty in the domains, where actions may fail or have unexpected results, scheduling activities are typically interleaved with planning and execution. Thus scheduler inefficiencies are multiplied many times during a problem solving instance.

In general the upper-bound on the number of possible schedules for a task structure containing m methods is $\sum_{i=0}^{m} \binom{m}{i} i!$ and the $\omega(2^m)$ and $o(m^m)$ combinatorics of our scheduling problem precludes using exhaustive search techniques for finding optimal schedules. Design-to-Criteria copes with these explosive combinatorics by satisficing with respect to the goal criteria and with respect to searching the solution space. This satisficing dualism translates into four different techniques that Design-to-Criteria uses to reduce the search space and make the scheduling problem tractable:

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. This is achieved by creating and identifying partial solutions that seem likely to meet the criteria and concentrating further development on these classes of partial solutions, pruning or ignoring other partial solutions that are deemed least probable to lead to "good" solutions.

Approximation Schedule approximations, called *alternatives*, are used to provide an inexpensive, but coarse, overview of the schedule solution space. Alternatives contain a set of unordered actions that can be scheduled (ordered) to achieve a particular task along with estimates for the quality, cost, and duration distributions that may result from scheduling the actions. Alternatives are inexpensive to compute as the complex task interactions are only partially considered and ordering, resource, and other constraints are ignored. The alternative abstraction space is used in conjunction with the criteria-directed focusing to build schedules from alternatives that are most likely to lead to good schedules.

Heuristic Decision Making We mentioned the high order complexity of our scheduling problem as a whole, but the action ordering scheduling problem suffers from similar combinatorics. Given a set of n actions to perform, i.e., an alternative with n methods to schedule, there are $n!$ orderings that must be considered and the $O(n!)$ expense precludes trying all possible orderings. We cope with this complexity using a group of heuristics for action ordering. The heuristics take into consideration task interactions, attempting to leverage positive interactions while avoiding negative interactions. They also consider resource limits, individual action deadlines, task deadlines, commitments made with other problem solving agents, and other constraints. The heuristic algorithm reduces the $O(n!)$ action ordering problem to linear or low-order 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 do not achieve the high-level task, or, achieve the high-level task but do not live up to quality, cost, duration, or certainty expectations set by the estimates contained in the alternatives. This can be caused by an overconstrained problem, but also by complex task interactions that are glossed over by the alternative approximation and not considered by the action ordering heuristics. A secondary set of improvement (Zweben *et al.* 1994) heuristics act as a safety net to catch the errors that are correctable. Again, this problem is potentially computationally expensive as the required fix may be achievable by any combination of the actions in the task structure and it is impossible to ascertain if a hypothetical fix will generate the desired result until it is fully scheduled. Thus this aspect of the scheduling algorithm is also heuristic and relies on abstraction and criteria-directed focusing to reduce the complexity.

Algorithmically, the scheduling process applies the techniques above as follows:

1. Recursively build alternatives for the top-level task node by creating the alternatives for its subtasks and so forth. At each task node, use criteria-directed focusing to determine what alternatives to propagate to the parent tasks.
2. Using criteria-directed focusing, select the most promising top-level alternative from the set of unscheduled alternatives and heuristically construct a schedule for it.
3. Examine the schedule and suggest improvements by creating new alternatives that contain the improvements and add them to the top-level alternative set.
4. Repeat steps 2 and 3 until the termination criteria is met.

Design-to-Criteria thus copes with computational complexity by using the client goal criteria to focus processing, reasoning with schedule approximations rather than complete schedules, and using a heuristic, rather than exhaustive, scheduling approach. This methodology is effective because several aspects of the scheduling problem are soft and amenable to a satisficing approach. For example, portions of the client goal specification (Wagner, Garvey, & Lesser 1997a) express soft client objectives or soft constraints. Solutions often do not need to meet absolute requirements because clients cannot know *a priori* what types of solutions are possible for a given task structure due to the combinatorics. Similarly, soft task interactions also represent soft constraints that can be relaxed, i.e., they can be leveraged or not depending on the situation. Finally, though the TÆMS scheduling problem is more complex than many traditional scheduling problems because of its representation of multiple approaches for task achievement, it is also more flexible. If we view the scheduling activity as a search process, typically there is a neighborhood of solutions that will meet the client's goal criteria and the lack of exhaustive search, i.e., search by focused processing and approximation, does not necessitate scheduling failure.

To illustrate the neighborhood and focusing concepts, consider a brief example. Say a hypothetical client is interested in schedules, for a moderately complex task structure, that trade-off quality and duration and is more interested in keeping duration toward the lower end of the spectrum than achieving maximum quality (60% emphasis on duration and a 40% emphasis on quality). Figure 3 displays the root-level alternatives (solution approximations) that are produced using the criteria-directed focusing mechanism. For this problem instance, 656 intermediate alternatives were constructed and 39 were initially generated at the top-level, although the actual number of alternatives shown in the graph is slightly larger as the heuristic improvement mechanism creates new alternatives as the schedule process iterates. In stark contrast to the economical alternative set generated using the focusing mechanism, Figure 4 displays the exhaustive root-level alternative set that results when the focusing mechanism is not used. In this case, 9106 alternatives were explored during processing and 4444 alternatives were generated at the root level. Note that the top-rated alternatives in each case have similar quality and duration characteristics and exhibit similar quality/duration trade-offs – keeping duration under control while achieving good/high quality. The exhaustive alternative generation case produced a larger set of reasonable candidate alternatives, but the focused case still found a significant number of reasonable alternatives. The schedules produced from the two different alternative set likewise have similar quality and duration characteristics.

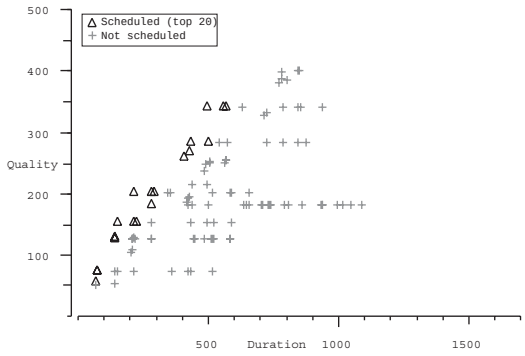

Figure 3: Top-Level Alternatives Produced by Focusing

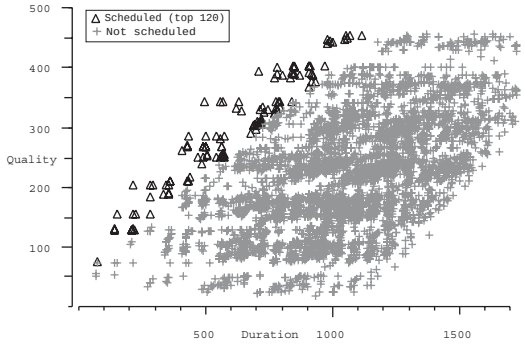

Figure 4: All Top-Level Alternatives

Research Applications

TÆMS and the Design-to-Criteria scheduler, and its predecessor, Design-to-Time (Garvey & Lesser 1995), are being used in a wide range of research projects including cooperative information gathering (Oates, Nagendra Prasad, & Lesser 1994; Decker *et al.* 1995), distributed collaborative design (Decker & Lesser 1995), and distributed situation assessment (Carver & Lesser 1995). The software architectures for the various research projects are related and based on a generic architecture consisting of scheduling, coordination, planning/problem solving, and execution/monitoring modules. The scheduler, coordination, and problem solving components interact by exchanging TÆMS task structures and the execution subsystem

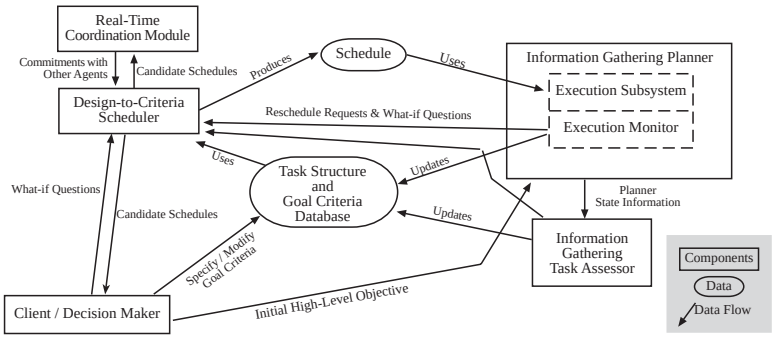

Figure 5: Information Gathering Agent Architecture

receives as input schedules augmented with TÆMS features. In our work, the coordination module *modulates* (Decker 1995a) the local scheduler-centric control mechanism rather than replacing it.

Cooperative information gathering is a multi-agent solution to search, discovery, and assimilation of information via the WWW. A typical information gathering task is to gather information to support a decision of whether to purchase WordPerfect, Microsoft Word, or WordPro, or some yet unidentified word processor. The project utilizes supporting technology from both the information retrieval (Callan, Croft, & Harding 1992) and information extraction (Fisher *et al.* 1996) disciplines to process retrieved documents. The inevitable advent of structured WWW data will also assist with the data interpretation task.

The instantiation of the generic agent architecture for the information gathering project is shown in Figure 5. In the cooperative information gathering project, the high-level information gathering task, the goal criteria, and a portion of the resource constraints originate with the decision maker module. The decision maker can be any high-level problem solving entity or agent, e.g., humans or controllers based on decision theoretic inference networks (Zilberstein & Lesser 1996). In this project, the execution subsystem and monitoring components are integrated with the problem solver. The problem solving functionality is provided by a RESUN (Carver & Lesser 1995) blackboard planner that explicitly represents sources-of-uncertainty associated with hypothesis and uses bottom-up and top-down processing to achieve goals and resolve sources-of-uncertainty. The RESUN planner is coupled with a with a *task assessor* component that examines the state of the blackboard and describes a portion of the available problem solving actions in the TÆMS language. The TÆMS task structure is then passed to the scheduler and the coordination mechanism for scheduling and coordination with other agents. The resulting schedule is passed back to the problem solver for execution and monitoring. As execution progresses and new evidence is found, or execution performance falls outside of the tolerance envelopes generated by the scheduler, replanning, rescheduling, and coordination occurs.

Meta-Analysis and Future Work

Design-to-Criteria produces custom schedules in the face of high-order complexity by satisficing with respect to the client goal criteria and with respect to the scheduling activity itself. Algorithmically, the satisficing methodology takes the form of criteria-directed focusing, approximation, heuristic decision making, and heuristic error correction. These techniques are effective and efficient.

There are two different facets to coping with complexity and resource constraints in soft real-time real-world problem solving systems. The first facet is constructing a course of action for a given set of resource constraints and goal criteria. In the TÆMS and Design-to-Criteria realm, this translates into custom building schedules as described in this paper. The second facet is determining the amount of resources to spend constructing the solution path or schedule versus the amount of resources to spend executing actions. Focused processing is central to our system and currently the focusing *degree* is static, predetermined by either the client or by default scheduler settings. This means that regardless of the complexity of the task structure or its size, the scheduler focuses to the same degree unless otherwise instructed by the client. While adequate, this approach is weak because the scheduling versus execution trade-offs are not reasoned about explicitly. In particular situations, with particular classes of task structures and goal criteria, it may be worthwhile to spend more resources building schedules, exploring a larger percentage of the solution space at the expense of execution because the schedules produced are likely to have much greater utility. In other situations, spending more resources scheduling may not yield higher utility in which case the resources are better spent executing actions. Future work will involve meta-analysis of problem instances, i.e., task structures, resource constraints, and goal criteria, to determine the focusing degree dynamically.

An area of future work related to meta-analysis is the refinement of the interface between the scheduler and other complex problem solving components and/or humans. In-

teractive 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 want to explore more of the search space or 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 and meta-analysis are clearly the next steps in exploiting and leveraging the power of the Design-to-Criteria paradigm.

References

Callan, J. P.; Croft, W. B.; and Harding, S. M. 1992. The INQUERY retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert Systems Applications*, 78–83.

Carver, N., and Lesser, V. 1995. The DRESUN testbed for research in FA/C distributed situation assessment: Extensions to the model of external evidence. In *Proceedings of the International Conference on Multiagent Systems*.

Dean, T., and Boddy, M. 1988. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 49–54.

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

Decker, K.; Lesser, V.; Prasad, M. N.; and Wagner, T. 1995. MACRON: an architecture for multi-agent cooperative information gathering. In *Proccedings of the CIKM-95 Workshop on Intelligent Information Agents*.

Decker, K. S. 1995a. *Environment Centered Analysis and Design of Coordination Mechanisms*. Ph.D. Dissertation, University of Massachusetts.

Decker, K. S. 1995b. TÆMS: A framework for analysis and design of coordination mechanisms. In O'Hare, G., and Jennings, N., eds., *Foundations of Distributed Artificial Intelligence*. Wiley Inter-Science. chapter 16.

Fisher, D.; Soderland, S.; McCarthy, J.; Feng, F.; and Lehnert, W. 1996. Description of the UMass Systems as Used for MUC-6. In *Proceedings of the 6th Message Understanding Conference*.

Garvey, A., and Lesser, V. 1995. Representing and scheduling satisficing tasks. In Natarajan, S., ed., *Imprecise and Approximate Computation*. Norwell, MA: Kluwer Academic Publishers. 23–34.

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*. Also UMASS CS TR–94–08.

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*.

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

Russell, S. J., and Zilberstein, S. 1991. Composing real-time systems. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, 212–217.

Wagner, T.; Garvey, A.; and Lesser, V. 1997a. Complex Goal Criteria and Its Application in Design-to-Criteria Scheduling. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*. Also available as UMASS Department of Computer Science Technical Report TR-1997-10.

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., and Lesser, V. 1996. Intelligent Information Gathering for Decision Models. UMASS Department of Computer Science Technical Report TR-96-35.

Zilberstein, S., and Russell, S. J. 1995. Optimal composition of real-time systems. *Artificial Intelligence* 79(2).

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

Zweben, M.; Daun, B.; Davis, E.; and Deale, M. 1994. Scheduling and rescheduling with iterative repair. In Zweben, M., and Fox, M., eds., *Intelligent Scheduling*. Morgan Kaufmann. chapter 8.