Multi-criteria decision quality optimization as a Scheduling problem

Abdel-Illah Mouaddib
CRIL-IUT de Lens-Université d'Artois
Rue de l'université, S.P. 16
62307 Lens Cedex, France
mouaddib@cril.univ-artois.fr

Abstract

Decision quality under uncertainty is a problem that has been addressed by many researchers. But most of the works dedicated to this problem assume that there is a single quality measure to computation based on the status of an n-tuple of criteria. However, in most of real-world applications the quality is multi-criteria where each criterion captures a dimension of value in the solution. In this paper, we present a decision maker that manipulates the quality as a vector of criteria where the improvement of each criterion is in charge of a progressive processing agent that uses its hierarchy of processing levels to improve the quality of this criterion incrementally. We discuss the problem of optimizing the decision of sequencing processing levels of agents by transforming it to an MDP. We also present the strategy to be adopted when considering the criteria preferences. Finally, we discuss how this approach can be applied to controlling a static set of dependent progressive processing agents where there is no predetermined sequence of agents.

Introduction

In real-world applications, the solution quality is frequently a function of multiple criteria. Computational utility is represented, in general, as a single utility measure to computation based on the status of an n-tuple of criteria. A set of criteria captures dimensions of value in a result, for example accuracy, precision and uncertainty. Criteria can be dependent in sense that the quality of a criterion can affect the quality of others. A sequence of computation action, a, applied to an initial problem, yields a result having a quality described as vector $\vec{Q}$ of quality criteria $(q_1, q_2, \ldots, q_n)$. Each action improves the quality of the result according to a subset of criteria. The system we work with uses a library of progressive processing agents (hierarchy of processing levels) each of which affects a criterion quality of the solution while the other criteria remain inchanged. We name this, *quality stability assumption*. The system has to optimize the multi-criteria decision under time pressure. This problem consists in finding the optimal sequence of processing levels of multiple progressive processing agents such that an optimal multi-criteria decision quality is constructed.

The existing approaches of controlling progressive processing (Mouaddib & Zilberstein 1997; 1998; Zilberstein & Mouaddib 1999) gives a substantial basic to develop a strategy to solve this problem. Indeed, the previous approaches can be seen as strategies of sequencing the processing levels agent by agent assuming the independency from the agents. However, this last assumption is not valid in our context since dependencies could exist between agents due to the fact that the state of a criterion can affect the improvement of others. In our approach, we introduce this dependency that leads to construct a sequence of processing levels independently of the sequence of agents. Differently speaking, processing level l^i of agent α can be followed by processing levels l^j of other agents β before sequencing its next level l^{i+1}. In the previous approaches, the sequence of processing levels has been constructed in respect to the sequence of agents that, in general, was constructed by the *earliest-deadline-first* order. The optimization of multi-criteria decision quality problem can be seen as a problem of controlling a *static* set of *dependent* progressive processing agents with the same deadline where no predetermined sequence of agents is defined.

An application of this approach could be the perception process of an autonomous robot where each sensor improves a criterion of the perception. For each amount of time the sensors have a quality of their sensing. Different sensors allow the robot to recognize its location in 3D. There is also a multi-criteria decision quality problem for a robot using sonar and infrared sensor since they contribute differently to improve the quality the perception task. Sonar allows to improve obstacle detection (first criterion) while infrared allows to improve distance estimation (second criterion). In this example, the problem is to determine the best sequence of acting with sonar and infrared to optimize the sensing operation of the robot. We present a multi-criteria decision approach where the quality stability is assumed. However, this assumption is not valid in some applications such the robotic application sketched above. The relaxation of this assumption does not affect the framework presented as it will be shown.

We can also apply this approach to dependent problems. Let assume the problem introduced in (Zilberstein & Russell 1993) where *sensing, planning* and *executing* phases can be seen as three dependent problems of a mobile robot. The predetermined order and complete operations of sensing, planning and execution don't guarantee to meet the optimal decision under resource constraints. However, it is more suitable to construct a sequence of partial operations of sensing, planning execution at any order until the available resource has been fully terminated. Our approach constructs an optimal policy that determines what partial operation (sensing, planning or execution) will be executed based on the actual progress of the other operations (sensing, planning and execution).

Section 2 presents progressive processing agents. Section 3 describes the formal framework of the approach. Section 4 presents the MDP controller. Section 5 discusses the same problem by considering preferences on criteria. Section 6 shows the analysis of the approach and its limits. Section 7 discusses how multi-criteria decision quality can solve the problem of controlling dependent agents. Section 8 presents the related work and finally we conclude the paper at section 9.

Progressive processing agents

Progressive processing (Mouaddib 1993) is a problem-solving technique that allows a system to trade-off solution quality against computational resources, sharing then the same motivation of resource-bounded reasoning techniques as *flexible computation* (Horvitz 1988), *anytime algorithms* (Boddy & Dean 1994; Zilberstein & Russell 1993), *imprecise computation* (Liu *et al.* 1991) and *design-to-time* (Garvey & Lesser 1993).

A progressive processing agent is a multi-level deliberation hierarchy based on processing levels L_α^i ordered by a linear precedence-constraint graph. Each level L_α^i can begin execution only after level L_α^{i-1} completes. Level L_α^i is the immediate successor of L_α^{i-1}. In general, the output of an agent can be used by any successive agent as soon as the first level has finished its processing. We assume that the information necessary for successive agents is produced by the first level. The hierarchical structure of each agent allows to characterize precisely the tradeoff between computation time and the quality of the result. The solution quality is discretized into a finite number of levels, $q_{L_\alpha^1}$, $q_{L_\alpha^2}, \ldots, q_{L_\alpha^{n_\alpha}}$, where $q_{L_\alpha^1}$ is the lowest quality and $q_{L_\alpha^{n_\alpha}}$ is the highest quality returned respectively by the first level L_α^1 and last level $L_\alpha^{n_\alpha}$. The purpose of monitoring is to optimize the time/quality tradeoff offered by each agent. This reasoning is especially suitable for domains where the reasoner uses abstraction to structure the search space (as in hierarchical planning), and for problems that require the result to be expressed at varying levels of detail (as in model-based diagnosis).

However the use of progressive processing as a component of a real-time system (Mouaddib & Zilberstein 1997; 1998; Zilberstein & Mouaddib 1999) introduces a problem of execution control: determining the sequence of component progressive processing levels such that the quality of the overall system result is maximized.

Optimizing the multi-criteria decision quality

Multi-Criteria quality representation

To capture the dimensions of value in a multi-criteria quality solution, we use a vector of criterion quality represented by n-tuple $(q_1^i, q_2^j, \ldots, q_n^k)$ where each quality criterion q_i^j represents the quality of the dimension i of the value in the solution quality after the execution of level l_i^j (j^{th} level of the agent i). Each criterion quality q_i^j is normalized ($q_i^j \in [0,1]$). Applications where the quality is not normalized we can use the percentage of improvement in order to use a measure that belongs to [0,1].

Performance profile

Each individual agent i has a characterisation of its performance that maps the status of an input quality vector $\vec{Q}$ to a discrete probability distribution of the duration t and output quality q_i^j that is the quality of dimension i in the vector after the execution of j^{th} level. This performance profile is denoted $PP_i((q_i^j, t)|\vec{Q})$. This performance profile is conditional as it is described in (Zilberstein & Russell 1993) and expresses the dependency between the status of the vector and the duration and the output quality of the processing level. The output quality can depend only on a subset of dimensions of the vector. This definition generalizes the definition of the conditional performance profile and allows to use a vector of dimensions to represent dependencies between each other. The performance profile PP allows also to express the dependency between criteria since the probability to get a quality of a criterion depends on the qualities of other criteria expressed in the vector.

Time-dependent utility function

The utility function $U(\vec{Q},t)$ of the output vector quality represents the utility of the status of the quality vector at time t. One way to define this utility is to take the distance between vector $\vec{Q}$ and vector $\vec{\mathbf{1}}$ representing the maximal quality of the solution and the cost of the time t. Vector $\vec{\mathbf{1}}$ is n-tuplet (1,1,...,1) while vector $\vec{\mathbf{0}}$ is n-tuplet (0,0,...,0)

$$U(\vec{Q},t) = -distance(\vec{Q},\vec{\mathbf{1}}) - Cost(t) \quad (1)$$

The function *cost* is the cost provoqued in the system when consuming t time units. Moreover, the distance function that define the euclidian measure is defined as follows:

$$distance(\vec{Q},\vec{\mathbf{1}}) = \sqrt{\sum_i (1-q_i)^2} \quad (2)$$

The distance between vector $\vec{Q}$ and vector $\vec{\mathbf{1}}$ is used to measure the rewarded value of the state by having a higher value when vector $\vec{Q}$ is close to vector $\vec{\mathbf{1}}$. In addition to the mathematicl convenience of the distance measure, it allows to prefer a vector over another when it is close to vector $\vec{\mathbf{1}}$ representing the highest quality (qualities has been normalised).

Meta-level control

The problem of the control consists in distributing available time T to progressive processing agents such that the multi-criteria decision is optimal given a probabilistic performance profile of agent processing levels and a utility time-dependency function. This problem can be seen as an MDP automaton where the states can be represented by the status of the vector and the time t consumed. The rewards associate with a state are the utility of the state and the possible action is to execute the next processing level of the agent.

Using MDP for the control

State representation The world of state is modeled as a stochastic automaton with a finite set of world states $\mathcal{S} = \{[\vec{Q},t]\}$

The initial state

$[\vec{\mathbf{0}},0]$

Terminal states

$[\vec{Q},T]$ and $[\vec{\mathbf{1}},t]$ that represent respectively the state where the available time has been fully elapsed or the state where the highest vector quality is reached.

Intermediate states

Intermediate states code the status of the vector at time t. These states have the following form: $[\vec{Q},t]$

Actions

In every nonterminal state the possible action $\mathbf{E}_i^j$ is to execute the level j of the agent i in order to improve the quality of the criterion i (dimension i in the value of the result). There is uncertainty on execution time and quality and quality dependency.

Transitions

The transition model is a function that maps each element of $\mathcal{S} \times \{\mathbf{E_i^j}\}$ into a discrete probability distribution over $\mathcal{S}$. In the following the transitions equations are described that code the transitions from an intermediate state to another and from an intermediate state to a terminal one.

Successful transition is when the action requires less resources than available. This transition is coded by:

$$if\ t + c_i^{j+1} \leq T,$$
$$Pr([\vec{Q'}, t + c_i^{j+1}] | [\vec{Q},t], E_i^{j+1}) = PP_i((q_i^{j+1}, c_i^{j+1}) | \vec{Q}) \quad (3)$$

The vector $\vec{Q'}$ corresponds to the new status of the vector of quality after the execution of the level $j+1$ of the agent i that leads to improve the criterion i (quality stability assumption) of the vector quality $\vec{Q}$. Other criteria can be improved if we relax the stability assumption. Vectors $\vec{Q'}$ and $\vec{Q}$ have the following forms:

$\vec{Q} = (q_1^x, q_2^y, \ldots, q_z^a, \ldots, q_i^j, \ldots, q_n^k)$

• *Quality stability assumption*

$\vec{Q'} = (q_1^x, q_2^y, \ldots, q_z^a, \ldots q_i^{j+1}, \ldots, q_n^k)$

• *Quality stability assumption relaxed*

$\vec{Q'} = (q_1^x, q_2^y, \ldots, q_z^{a+1}, \ldots, q_i^{j+1}, \ldots, q_n^k)$

When the quality stability assumption is relaxed, vector $\vec{Q'}$ is the result of the modification of the qualities of more than one quality criterion of vector $\vec{Q}$. The rest of the discussion is not affected by the quality stability assumption.

Failure transition is when the action requires more resources than available. In such situations, the mechanism consists of aborting the action.

$$if\ t + c_i^k > T,$$
$$Pr([\vec{Q},T] | [\vec{Q},t], E_i^j) = \sum_{n \geq k \leq j+1 \ and \ \forall q'} PP_i((q', c_i^k) | \vec{Q}) \quad (4)$$

Value and Reward

The value of intermediate states

$$V([\vec{Q},t]) = \max_i$$
$$(PP_i((q_i^{k+1}, c_i^{k+1}) | \vec{Q}) V([\vec{Q'}, t + c_i^k]) +$$
$$\sum_{\forall c_i^{k+1} \ c_i^{k+1}+t>T, \ \forall \ q_i^{k+1}} PP_i((q_i^{k+1}, c_i^{k+1}) | \vec{Q}) V([\vec{Q},T])) \quad (5)$$

The value of terminal state

$$V([\vec{1},t]) = U(\vec{1},t) = -Cost(t) \quad (6)$$

$$V([\vec{Q},T]) = U(\vec{Q},T) \tag{7}$$

The MDP obtained is a finite-horizon MDP with no cycles, because we move to the next level per agent, the transitions move "forward" with no loops. This MDP can be easily solved for a reasonable size (relatively large state spaces) using standard dynamic programming algorithms or search algorithm AO^*.

Theorem 1 *The optimal multi-criteria decision quality is the optimal policy computed for its corresponding MDP.*

Proof:
Because of the one-to-one correspondence between the state of the MDP and the computation state of the multi-criteria decision quality problem, the optimal solution of the MDP is the optimal multi-criteria decison quality. $\Box$

Considering preferences on criteria

We have discussed the problem of sequencing multiple progressive processing agents for multi-criteria decision quality, in a general case. However, in this section we discuss a specific policy for sequencing multiple progressive processing agents where criteria are ordered according to their preference. The preference definition we use in our context is given as follows:

Definition 1 $i \succ j$ *means that:* $\forall \vec{Q}$ *and* $\vec{Q'}$, *if* $q_i > q'_i$ *then we have,*

$$\forall t,t' \;\; Pr([\vec{Q},t])V([\vec{Q},t]) > Pr([\vec{Q'},t])V([\vec{Q'},t'])$$

In this definition we introduce the probability to be in a state, Pr($[\vec{Q}$,t]), that we show how it is computed in our context. The intuitive sense we give to this definition is that the strategy prefers a criterion over another when the quality of the first one is higher than the second and the probility to get this quality is higher than the probability to get the quality of the other criterion.

Theorem 2 *In situations where there is a preference function $\succ$ over criteria, the best policy is to allocate all the time to the most preferred criterion until its saturation (q=1), then the next most preferred and so on until the available time has been fully elapsed. This strategy is optimal.*

Proof:
We assume that $q_1 \succ q_2 \succ q_3 \ldots \succ q_n$ corresponding to the preference order on criteria $1, 2, \ldots, n$. Let π be the policy of the theorem.

Definition 2 *We say that a policy π is optimal if,* $\forall$ π' *and* $[\vec{Q},t]$ $V_\pi([\vec{Q},t]) > V_{\pi'}([\vec{Q},t])$.

Let assume that current state is $[\vec{Q},t]$. We can then compute the probability to be in state $[\vec{Q'},t+c_i^j]$ when moving from current state $[\vec{Q},t]$ to $[\vec{Q'},t+c_i^j]$ as follows:

$$Pr([\vec{Q'},t+c_i^j]) = Pr([\vec{Q'},t+c_i^j]|[\vec{Q},t],\mathsf{E}_i^j) \tag{8}$$

Because of the equations 3 and 4, the above equation becomes :

$$Pr([\vec{Q'},\min(t+c_i^j,T)]) = \sum_{\forall(q',c_i^j)} Pr((q_i^j,c_i^j)|\vec{Q}) \tag{9}$$

Let assume, for contradiction, that from state $[\vec{Q},t]$ we use another policy π' rather than policy π of the theorem such as $\pi' \neq \pi$.

We execute agent j by using policy π'. Action a' is the action indicated by policy π' at current state $(\pi'([\vec{Q},t]))$. The value with this transition is:

$$V_{\pi'}([\vec{Q},t]) = \sum_{\forall(q_j,\delta)} PP_{a'}((q_j,\delta)|\vec{Q})V_{\pi'}([\vec{Q''},\min(t+\delta,T)]) \tag{10}$$

The transition with policy π is that we execute agent i where action a is the one indicated by policy π at current state $(\pi([\vec{Q},t]))$:

$$V_\pi([\vec{Q},t]) = \sum_{\forall(q_i,\delta)} PP_a((q_i,\delta)|\vec{Q})V_\pi([\vec{Q'},\min(t+\delta,T)]) \tag{11}$$

Because criterion i is preferred to j and that in equation (11) the quality of this criterion is improved $(q'_i > q_i)$ and that in equation (10) criterion i is not modified, we have: $q'_i > q''_i$. Moreover because of the definition 2 we can say that:

$$\begin{aligned} &Pr([\vec{Q''},\min(t+\delta,T)])V_{\pi'}([\vec{Q''},\min(t+\delta,T)]) \quad < \\ &Pr([\vec{Q'},\min(t+\delta,T)])V_\pi([\vec{Q'},\min(t+\delta,T)]) \end{aligned} \tag{12}$$

Because of the equation 9 we have:

$$\begin{aligned} &\textstyle\sum_{\forall(q_i''^j,\delta)} Pr((q_i''^j,\delta)|\vec{Q})V_{\pi'}([\vec{Q''},\min(t+\delta,T)]) \quad < \\ &\textstyle\sum_{\forall(q_i'^j,\delta)} Pr((q_i'^j,\delta)|\vec{Q})V_\pi([\vec{Q'},\min(t+\delta,T)]) \end{aligned} \tag{13}$$

Because of equations 10, 11 and 13 we can say that:

$$V_{\pi'}([\vec{Q},t]) < V_\pi([\vec{Q},t]) \tag{14}$$

The maximisation of the equation (5) has not been respected. Contradiction. $\Box$

Analysis

The number of states of the MDP is defined by n the number of criteria, k be the maximum number of processing levels per agent and l is the maximum number of alternative in PP_a that are described by the performance profile. The number of states is, then, bounded by ln^k. An example of 10 criteria, 5 levels and 10 alternatives per PP_a leads to 10^6 states of MDP.

We have implemented the policy construction algorithm using standard dynamic programming algorithms. We used a library of PP_a constructed off-line for each action. This construction is based on a statistical data gathered during excutions of actions. We use a discret variable representing PP_a. This representation is based on a set of quadruple (q_{out}, t, Q_{in}, p), where p is the probability to get the output quality q_{out} of the criterion and allocating time t when the current state of the quality criteria is Q_{in}.

The feasibility of the approach is highly related to the problem of constructing PP_a and precision in representing it. Indeed, the set representing PP_a affects the precision since the representation is more precise with more elements in the set. However, with a large set of PP_a the computation time to generate the policy becomes high. The experiments consist in evaluating the benefit rewarded in the policy returns according to the precision on PP_a (larger set). The policy perfromance is also affected by the number of criteria since it is obvious that the size of the MDP increases with the number of criteria. The experimental results are under development.

Multi-Criteria decision quality optimisation and the control of dependent agents

The problem addressed in this paper can solve the issue of controlling dependent agents with the same deadline where no sequence is predetermined. Indeed, execution of agents could consist of a partial execution of an agent followed by a partial execution of others before pursuing its execution. Furthermore, each partial execution can affect the progress of the execution of the other agents.

In the autonomous robot problem introduced in (Zilberstein & Russell 1993), the approach adopted consists of a sequence of sensing, planning and execution respecting the order of those operations and studing how to allocate time to those operations so as to maximize the overall quality. In our approach, the order of sensing, planning and execution is not required and those operations can be done at any order as for example: $\mathsf{s}_1, \mathsf{s}_2, \mathsf{p}_1, \mathsf{e}_1, \mathsf{s}_3, \mathsf{p}_2, \mathsf{p}_3, \mathsf{s}_4, \mathsf{e}_2, \ldots$, where o_i is the i^{th} step of the operation o (s for sensing, p for planning and e for execution). Indeed, a partial execution can allow a better improvement of the quality of a partial sensing that can contribute to obtain a better quality improvement of a partial planning. The order of those operations is guided by a conditional performance profile PP_a.

More formally speaking, let $\{P_1, P_2, \ldots, P_n\}$ be the static set of problems to solve in an available time T. Problems P_i are dependent on each other and there is no predetermined sequence. Let a_i be agent i that gradually solve problem P_i (agent for sensing and data interpretation for example). This problem can be represented as a multi-criteria decision quality since we can use vector $\vec{Q} = (q_1, q_2, \ldots, q_n)$ of solution qualities of problems P_i where q_i is the solution quality of problem P_i. The dependency between problems can be represented by conditional performance profile PP_a that describes the distribution probability on the solution quality of a problem given the solution qualities of the other problems including the quality of this problem. We easily formalize the quality dependency between problem that can also formalize the precedence constraint between problems.

Definition 3 *We say that problem p depends on problem q when we have to solve, even partially, first problem q before solving problem p. We say that problem q preceeds problem p.*

We formalize this definition by using PP_a such that:

$$PP_p((Q_p = 0, t)|Q_q = 0) = 1 \tag{15}$$

We can generalize this equation for situations where problem p depends on problems $r, s, \ldots, u$ such that:

$$\vec{Q} = (q_1, \ldots, q_r, q_s, \ldots, q_u, \ldots, q_n)$$
$$\forall\, \vec{Q},\ q_r = 0 \; or \; q_s = 0 \; or \; q_u = 0, PP_p((Q_p = 0, t)|\vec{Q}) = 1 \tag{16}$$

This equation formalizes that the quality of the solution of the problem p is equal to 0 when all the problems on which p depends are not (partially) solved.

Related Work

This paper shares the same motivation of substantial works that use dynamic programming to solve the problem of meta-level control of resource-bounded reasoning. Hansen and Zilberstein (Hansen & Zilberstein 1996) have used the dynamic programming for monitoring the progress of anytime algorithms. The sequencing contract algorithms problem (Zilberstein, Charpillet, & Chassaing 1999) has also been studied for controlling a contract algorithm with a stochastic deadline. Another approach has been studied in (Delhay, Mouaddib, & Zilberstein 1999) by considering the problem of sequencing multiple contract algorithms. This approach doesn't deal with dependency of problems. Optimal monitoring of dynamic progressive processing has also been studied in (Mouaddib & Zilberstein 1998) without the problem dependency and the quality dependency and uncertainty. A reactive control of progressive processing has been studied in (Zilberstein & Mouaddib 1999) where a richer and non-linear structure of progressive processing is presented considering duration uncertainty and quality dependency and uncertainty. This approach considers agents one by one and assumes that the execution of a level can not be conditioned by another level of another agent (agents dependency). Our approach can be applied to a control problem of a static set of dependent problems with the same deadline and allows to relax the assumption of independency between problems. Moreover, our approach of optimisation of multi-criteria decision quality can be applied to the problem of combining dimensions addressed in any-dimension approach (Musliner, Durfee, & Shin. 1992) where any-dimension methods lead to guarantee dimensions (time, memory, certainty, accurancy, ...). Our approach construct a policy to show how any-dimension methods can be executed under resource limit by maximizing the rewarded value of the combined dimensions result. Design-to-criteria (Wagner, Garvey, & Lesser 1998) is another approach of multi-criteria decision quality where the methods allow to construct schedules each of which meets a specific design criterion. The construction of the schedule is guided by a specific goal criterion. In other words, this approach makes some criteria preferred to others (that will be ignored) and construct a decision that maximize the quality of selected criteria. In our approach, we determine the optimal sequence of improving all criteria by determining which criterion will be improved based on the actual progress of the other criteria qualities (the status of the vector). This approach , in addition to handle duration uncertainty, dependency and uncertainty quality, handles agent dependency. However, this approach seems to be unsuitable for problems where the set of criteria changes dynamically from one situation to another.

Conclusion

We present a new approach to construct an optimal multi-criteria decision quality by sequencing processing level of multiple progressive processing agents. This approach is highly related to the approaches previously studied for monitoring progressive processing by considering the dependency between agents. The optimal multi-criteria decision quality problem has been reformulated by an MDP, a vector of dimensions (the quality criterion) and we then show the construction of an optimal policy. We also show that when considering the criteria preferences the optimal policy consists of sequencing the processing levels in respect to the sequence of agents that is defined by the order of criteria preference. We have also discussed how this approach can be used to control dependent agents where no predetermined sequence is available.

Future works will be considered for an experimental evaluation and showing the reasonable size of the MDP and how this approach can be applied to the previous problems by considering the dependency between problems. We will also guide our future work on the problem of a dynamic set of criteria and how to construct PP_i. The performance profile PP_i in our context is derived from an average of different states of the quality vector. Because the size of the quality vector will be modified dynamically a new approach should be developped for constructing the PP_i. This issue concerns the construction of the performance profile in dynamic environments.

References

Boddy, M., and Dean, T. 1994. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence* 67:245–285.

Delhay, A.; Mouaddib, A.-I.; and Zilberstein, S. 1999. Sequencing multiple contract algorithms. In *IJCAI-99 Workshop on Planning and Scheduling Meet Monitoring in Dynamic and uncertain environment*, 79–83.

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

Hansen, and Zilberstein. 1996. Monitoring the progress of anytime problem-solving. *AAAI-96* 1229–1234.

Horvitz, E. 1988. Reasoning under varying and uncertain resource constraints. In *AAAI-88*, 111–116.

Liu, J.; Lin, K.; Shih, W.; Yu, A.; Chung, J.; and Zhao, W. 1991. Algorithms for scheduling imprecise computations. *IEEE Transactions on Computer* 24(5):58–68.

Mouaddib, A.-I., and Zilberstein, S. 1997. Handling duration uncertainty in meta-level control of progressive processing. In *IJCAI-97*, 1201–1206.

Mouaddib, A.-I., and Zilberstein, S. 1998. Optimal scheduling for dynamic progressive processing. In *ECAI-98*, 499–503.

Mouaddib, A.-I. 1993. *Contribution au raisonnement progressif et temps réel dans un univers multi-agents.* PhD, University of Nancy I, (in French).

Musliner, D.; Durfee, E.; and Shin., K. 1992. Any-dimension algorithms. In *Proc. Workshop on Real-Time Operating Systems and Software.*

Wagner, T.; Garvey, A.; and Lesser, V. 1998. Criteria directed task scheduling. *Special Scheduling Issue of the International Journal of Approximate Reasoning* 19:91–118.

Zilberstein, S., and Mouaddib, A.-I. 1999. Reactive control for dynamic progressive processing. In *IJCAI-99*.

Zilberstein, S., and Russell, S. 1993. Anytime sensing, planning and action: A practical model for control robot. In *IJCAI-93*, 1402–1407.

Zilberstein, S.; Charpillet, F.; and Chassaing, P. 1999. Real-time problem solving with contract algorithms. In *IJCAI-99*.