Decision-Theoretic Control of Inference for Time-Critical Applications

Thomas Dean

Brown University
Dept. of Computer Science
Providence, Rhode Island 02912

Technical Report No. CS-89-44
Revised April 1990

Decision-Theoretic Control of Inference for Time-Critical Applications

Thomas Dean*
Department of Computer Science
Brown University
Box 1910, Providence, RI 02912

Abstract

Systems that operate successfully in the real world have to coordinate their behavior with the behavior of processes outside their immediate control. Since the decision making required for good performance takes time and expends computational resources, it is often useful for a system to reason about its decision-making capabilities. This paper surveys a number of approaches for designing systems capable of managing their computational resources so as to maximize the expected utility derived from decision making. In particular, we consider methods that enable a system to trade some measure of decision-making accuracy for speed in problem solving and planning. In some cases, the tradeoffs can be made at design time and compiled into a run-time system. Alternatively, general knowledge for making the tradeoffs can be built into the system enabling it to make the necessary tradeoffs at run time. We explore the issues involved in making both sorts of tradeoffs and provide directions for further research.

*This work was supported in part by a National Science Foundation Presidential Young Investigator Award IRI-8957601 with matching funds from IBM, and by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-88-C-0132.

1 Introduction

This paper provides a survey of approaches for designing systems that are capable of taking their own computational resources into consideration during planning and problem solving. In particular, we are interested in systems that manage their computational resources by using expectations about the performance of decision making procedures and preferences over outcomes resulting from applying such procedures. Careful management of computational resources is important in complex problem solving tasks in which the time spent in decision making affects the quality of the responses generated by a system.

Much of the work described in this paper can be seen as a response to a movement, started in the early 1980's, away from systems that make use of complex representations and engage in lengthy deliberations, and towards systems capable of making many very simple decisions quickly. This movement brought about the advent of the so-called "reactive systems." Most reactive systems are essentially programming languages for building systems that must be responsive to their environment. Such languages generally allow for multiple asynchronous decision processes, facilitate communication among processes, and provide support for interrupts and process arbitration.

Many of the researchers building reactive systems were interested in robotics and decision-support applications requiring real-time response. The responsiveness of reactive systems was in stark contrast with the performance of most planning and problem solving systems in use at that time. Most existing planning systems were essentially off-line data processing procedures that accepted as input some initial (and generally complete) description of the current state of the environment, and, after some indeterminate (and generally lengthy) delay, returned a rigid sequence of actions which, if the environment was particularly cooperative, might result in the successful achievement of some goal.

Reactive systems might be seen as an extreme response to the shortcomings of the existing planning systems. Reactive systems provided responsiveness at the cost of shallow and often short-sighted decision making. Since there were no proposals for how to control decision making in time-critical situations, researchers turned away from the traditional approaches to planning and attempted to incorporate more sophisticated decision making into reactive systems. Unwilling to sacrifice response time, the researchers that were trying to improve the decision making capabilities of reactive systems were forced to trade space for time, often without a great deal of attention to the consequences.

Some of the dissatisfaction with complex representations and complicated deliberation strategies was due to misinterpreting asymptotic complexity results as evidence of the existence of impassable computational barriers. Proofs of NP-hardness certainly indicate that we must be prepared to make concessions to complexity in the form of tradeoffs. The lesson to be learned, however, is that we have to control inference, and not that we have to abandon it altogether.

In the 1970's, a great deal of effort was spent studying systems capable of explicitly reasoning about their own decision-making capabilities [10, 11, 16, 17, 22, 48]. This sort of reasoning about reasoning is generally referred to as *meta-reasoning.* As the research in this area matured, some researchers were concerned with how to learn to control decision making [37, 39], while others were interested in the basic mechanisms required to guide decision making under time pressure. Many of the mechanisms studied had in common the use of expectations regarding the performance of decision procedures to help in selecting from among a set of such decision procedures.

As researchers began looking in the literature, it became clear that many of the tools required for reasoning about the costs and benefits of applying decision-making routines were already available. Indeed, researchers in the decision sciences had already considered some of the problems involved in reasoning about the costs and benefits of inference [18, 30, 41]. However, with rare exception,[1] the decision analysts assumed that the agent was possessed of unlimited computational capabilities for reasoning about its current knowledge; the issue most often addressed concerned whether or not an agent should consider adding to its current knowledge. We are interested in the case of an agent currently biased to act in a certain way and considering if it should expend further computational resources and risk the consequences of delay in order to deliberate further about its options. It is this basic idea of an agent with limited computational capabilities, embedded in a complex environment with other agents and processes not under its control, and reasoning about the costs and benefits of continued deliberation that is the subject of this paper.

[1] I. J. Good was one of those exceptions, and, in an amazingly forward looking paper [23], Good talked about what he called *type II rationality* which involves an agent reasoning about its own abilities to reason.

2 Decision Theory and the Control of Inference

We begin with the idea of a *decision procedure*: a procedure used by an agent to select an action which, if executed, changes the world. Some actions are purely computational. For our purposes, such computational actions correspond to an agent running a decision procedure, and we will refer to such actions as *inferential*. The results of inferential actions have no immediate effect on the world external to the agent, but they do have an effect on the internal state of the agent. In particular, inferential actions consume computational resources that might be spent otherwise, and generally result in the agent revising its estimations regarding how to act in various circumstances. In addition to the purely computational actions, there are *physical actions* that change the state of the world external to the agent, but that may also require some allocation of computational resources. Real agents have severely limited computational capabilities, and, since inferential actions take time, an inferential action may end up selecting a physical action that might have been appropriate at some earlier time, but that is no longer so. Inferential actions are useful only insofar as they enable an agent to select physical actions that lead to desirable outcomes.[2] Decision theory provides us with the language required to talk precisely about what it would mean for an action to lead to a desirable outcome. Before we can proceed further, we will need some precise language for talking about possible outcomes and stating preferences among them. The language that we adopt is borrowed directly from statistical decision theory [2, 9, 41].

In the simplest case, we might consider an agent faced with choosing from among a set of completely defined and immediately attainable alternatives (*e.g.*, a student might be faced with choosing between seeing a movie or studying for an exam). The agent might ignore some of the implications of its actions and focus on immediate rewards (*e.g.*, a relaxing respite from work or an increase in knowledge about a given subject), but, more often than not, the agent will be concerned with the long-term implications or *consequences* of its actions (*e.g.*, the possibility of achieving a high score on an exam which in turn might raise the chances of getting into graduate school). In general, we cannot guarantee these consequences; they are seldom immediately attainable and they are usually only partially defined. If we ignore the long-term implications of our actions, the alternatives

[2]Inferential actions are also useful for learning purposes, as in learning search strategies, but even here the actions are ultimately in service to selecting physical actions that lead to desirable outcomes.

can be viewed as rewards, and a rational agent would simply choose the reward that it considers best. In the case in which the agent is concerned about the consequences of its actions and those consequences are not entirely under its control, the picture is more complicated. In this case, the agent might have a probability distribution over the set of possible consequences and some way of assigning values to the individual consequences so as to form expectations regarding the value of the possible outcomes or *prospects* resulting from performing alternative actions.

Let Ω correspond to a set of possible states of the world. We assume that the agent has a function, $U : \Omega \rightarrow \Re$, that assigns a real number to each state of the world. This is referred to as the agent's *utility function.*[3] These numbers enable the agent to compare various states of the world that might result as a consequence of its actions. It is assumed that a rational agent will act so as to maximize its utility. The quantity, $U(\omega)$ where $\omega \in \Omega$, is generally meant to account for both the immediate costs and benefits of being in the state ω and the delayed costs and benefits derived from future possible states. We assume that there is some process deterministic or stochastic governing the transition between states, and that this process is partially determined or biased by the agent's choice of action. In the case of a stochastic process, the agent cannot know what state will result from a given action and hence the agent must make use of expectations regarding the consequences of its actions. In order to account for these longer-term consequences, it is often useful to think of the agent as having a particular long-term plan or *policy.* In such cases, the agent will generally assign an expected utility to a given state based upon the immediate rewards available in that state and expectations about the subsequent states, given that the agent continues to select actions based upon its current policy.

In addition to expectations about the possible future consequences of its physical actions, an agent capable of reasoning about its computational capabilities must also have expectations regarding the potential value of its computations, and estimates of how long those computations are likely to take. In most of the work discussed in this paper, an agent is assumed to engage in some sort of meta-reasoning. For our purposes, meta-reasoning consists of running a decision procedure whose purpose it is to determine what other decision procedures should run and when. We prefer the term *deliberation scheduling* [13] to the more general meta-reasoning and will use the two interchangeably in this paper. If the

[3]See Chernoff and Moses [9], Barnett [2], or Pearl [40] for discussions regarding the axioms of utility theory.

meta-level decision procedure takes a significant amount of time to run, it must be argued that this time is well spent. In some cases, the time spent in meta-reasoning is small enough that it can be safely ignored; in other cases, it may be useful to invoke a meta-meta-level decision procedure to reason about the costs and benefits of meta-reasoning.

The rest of this paper is organized as follows. We begin in Section 3 with a discussion of a theory about the value of information, and point out some of the problems that arise in trying to apply this theory to agents with limited computational capabilities. In Section 4, we consider two papers, one of which outlines a general approach to studying the control of reasoning, and a second that builds on this approach and explores some particular issues in detail. In this same section, we also investigate some of the practical issues that constrain how an agent might reason about its computational capabilities; these constraints and the measures taken to deal with them apply to all of the work discussed in this paper. Section 5 considers an approach to reasoning about computational capabilities that relies on a particular class of algorithms for implementing decision procedures. Section 6 briefly considers some related issues in design-time meta-reasoning for compiling run-time systems for time-critical applications.

3 The Value of Information

In the real world, information costs. Every time that you get operator assistance in dialing a long-distance number or consult an accountant about your income tax you are paying for information. It is often useful to be able to assess the value of information so as to make reasonable decisions regarding whether or not to pay for it. In this section, we consider a theory due to Howard [30] concerned with quantifying the value of information in decision-theoretic terms.

Suppose that we have calculated an expectation regarding the profit, v, we are to obtain from a particular business venture. We denote this expectation as

$$E[v|\mathcal{E}],$$

where the conditioning event $\mathcal{E}$ represents our current state of (background) knowledge. This expectation is computed by selecting actions that will serve to maximize expected utility based on the information at hand. Suppose that, prior to embarking on this venture, someone offers to sell us information pertaining to some variable, x, that we used

in calculating $E[v|\mathcal{E}]$. To be more specific, suppose that our informant is clairvoyant and knows the actual value of x. Let C_x correspond to the event of obtaining the information from the clairvoyant. The expected value of obtaining this information, v_{C_x}, is given by

$$E[v_{C_x}|\mathcal{E}] = E[v|C_x, \mathcal{E}] - E[x|\mathcal{E}]. \tag{1}$$

To compute $E[v|C_x, \mathcal{E}]$, we evaluate the expected profit given knowledge about x for each possible value of x provided by the informant, summing over these expectations weighted by our prior on x

$$E[v|C_x, \mathcal{E}] = \int_{\Omega_x} E[v|x, \mathcal{E}] \Pr(x|\mathcal{E}). \tag{2}$$

It is important to note, as did Howard in the 1966 paper [30] in which he introduced Equations 1 and 2, that we use the prior distribution $\Pr(x|\mathcal{E})$ for x because, until the informant provides the information about x, our knowledge of x is based entirely on our background knowledge $\mathcal{E}$.

There is more to information value theory than presented here.[4] In a later paper [31], Howard describes how the availability of information affects not just the expectation of profit, but the probability density function of profit, or profit lottery. Our brief discussion of information value theory should suffice, however, to point out some complications that arise in applying the theory to reasoning about systems with limited computational capabilities.

Note that so far we have not accounted for the computational cost of *deliberating* about the value of a particular information source. In information processing systems, information costs in terms of the time and resources expended in computing an answer to a query. Neither have we closely considered how an agent might compute an expectation such as $E[v|\mathcal{E}]$. We may know how to compute such an expectation, but it may be that an agent can not afford to compute it. In the following sections, we build on the basic idea behind information value theory to account for systems that have limited computational capabilities.

[4] It should be noted that Howard's is not the only theory proposed for assessing the value of information sources. In particular, information value theory is closely related to the theory of experimental design [20, 38]. Experimental design is concerned with the problem of maximizing the information gained from performing experiments under cost constraints. Information value theory represents one approach to experimental design based on Bayesian decision theory.

4 Decision-Theoretic Control of Problem Solving

In this section, we consider a general approach to reasoning about decision-theoretic control of inference due to Russell and Wefald [43, 44]. As in most decision problems, the basic goal is for the agent to maximize its utility function $U(\omega)$ on states of the world $\omega \in \Omega$. We assume that the agent has some set of *base-level* actions $\mathcal{A}$ that it can execute to affect its environment. Borrowing Russell and Wefald's notation, we denote the outcome of an action A performed in state ω as $[A, \omega]$ or just $[A]$ if the action is performed in the current state.

At any given time the agent has a default action $\alpha \in \mathcal{A}$ which is the action that currently appears to be best. In addition, the agent has a set of computational actions $\{S_i\}$ which might cause the agent to revise its default action. The agent is faced with the decision to choose from among the available options: $\alpha, S_1, S_2, \ldots, S_k$. Computational actions only affect the agent's internal state. However, time passes while the agent is deliberating and opportunities are lost, so the net value of computation is defined to be the difference between the utility of the state resulting from the computation minus the utility of the state resulting from executing α:

$$V(S_j) = U([S_j]) - U([\alpha]).$$

If the computation S_j results in a revised assessment of the best action, α_{S_j}, and a commitment to perform this action, then

$$U([S_j]) = U([\alpha_{S_j}, [S_j]]),$$

where $[\alpha_{S_j}, [S_j]]$ indicates the outcome of the action α_{S_j} in the state following the computation S_j. Alternatively, if S_j is a partial computation (*i.e.*, a computation that doesn't immediately result in a revised assessment of a best action, but that provides intermediate results leading to a revised assessment), then

$$U([S_j]) = \sum_T \Pr(T) U([\alpha_T, [S_j.T]]),$$

where T ranges over all possible complete computations following S_j, $S_j.T$ denotes the computation corresponding to S_j immediately followed by T, and $\Pr(T)$ is the probability that the agent will perform the computation T.

Generally, the agent doesn't know the exact utilities or probabilities, and so it must compute an estimate using some amount of its computational resources. Let $\hat{Q}^S$ denote the estimate of the quantity Q following a computation S. In this case, we have

$$\hat{U}^{\mathsf{S}.S_j}([S_j]) = \sum_T \hat{\Pr}^{\mathsf{S}.S_j}(T)\hat{U}^{\mathsf{S}.S_j}([\alpha_T, [S_j.T]]),$$

where S is the total computation prior to considering S_j, and

$$\hat{U}^{\mathsf{S}.S_j}([\alpha_T, [S_j.T]]) = \max_i \hat{U}^{\mathsf{S}.S_j}([A_i, [S_j.T]]),$$

where the A_i range over all possible base-level actions in $\mathcal{A}$. By superscripting quantities to indicate the computations required to generate the corresponding estimates, Russell and Wefald are attempting to capture the behavior of real agents with realistically limited computational capabilities. At each point in time, the agent decides how to act based upon whatever estimates it currently has, using a meta-reasoning decision procedure whose time cost is assumed to be negligible. The meta-reasoning decision procedure is responsible for deciding whether further deliberation is warranted, and it does so on the basis of the estimated net value of computation

$$\hat{V}^{\mathsf{S}.S_j}(S_j) = \hat{U}^{\mathsf{S}.S_j}([S_j]) - \hat{U}^{\mathsf{S}.S_j}([\alpha]).$$

As Russell and Wefald point out, before the computation S_j is performed $\hat{V}(S_j)$ is just a random variable, and so the agent, not knowing the exact value, computes an expectation

$$E[\hat{V}^{\mathsf{S}.S_j}(S_j)] = E[\hat{U}^{\mathsf{S}.S_j}([S_j]) - \hat{U}^{\mathsf{S}.S_j}([\alpha])]. \tag{3}$$

It is worth noting the difference between Equation 3 and Equation 1 in Howard's value of information theory. The important difference is that both terms in the right-hand-side expectation in Equation 3 change as a consequence of further inference. If an agent had unlimited computational capabilities, it would not be computing estimates, and only the first of the two terms would require expending computational resources since it would be the case that

$$\hat{U}^{\mathsf{S}.S_j}([\alpha])] = \hat{U}^{\mathsf{S}}([\alpha])].$$

However, we are concerned with agents with limited computational capabilities, and further computation will likely result in a better estimate of the utilities for $[\alpha]$ as well as for $[\alpha', [S_j]]$ for any action $\alpha' \in \mathcal{A}$.

In order to simplify reasoning about the utility of combined computational and base-level actions, Russell and Wefald separate the *intrinsic utility*, that is the utility of an action independent of time, from the *time cost* of computational actions, defining the utility of a state as the difference between these two:

$$\hat{U}([A_i, [S_j]]) = \hat{U}_I([A_i]) - TC(S_j).$$

It should be noted, however, that determining an appropriate time cost function can become quite complicated in applying Russell and Wefald's approach. In particular, costs concerning hard and soft deadlines will have to be accounted for by this function. In the game-playing application explored in [43], there are no hard deadlines on a per-move basis, instead there is a per-game time limit that is factored into the time cost. In many time-critical problem solving applications, calculating the time cost can be quite complicated (*e.g.*, consider the sort of medical care applications investigated in [29] and [26]). Russell and Wefald assume that the time cost is independent of both the computation itself and its recommendations. The former is certainly reasonable, and, since the recommendation is not known at meta-reasoning time, the latter is also reasonable. However, one could easily imagine employing an expected time cost based on some *a priori* knowledge concerning possible recommendations. It should be noted that all of the approaches described in this paper make assumptions about time cost similar to those of Russell and Wefald.

Perhaps the nicest part of the Russell and Wefald work is their careful treatment of the criterion for deciding whether or not to expend further resources on deliberation. If the agent is considering at most one additional computational step, then it is only interested in computations that serve to update the expected value of a given base-level action so as to supplant the current default action. The expected gain from a given computation is measured in terms of the difference between the current expectation regarding one action and the anticipated revised expectations regarding a second action where one of the two actions is the default action. Intuitively, further deliberation is called for whenever the difference between the expected gain in utility from a computation and the associated cost of delay is greater than zero. Russell and Wefald identify two cases to consider in deciding to perform a computation aimed at providing a revised assessment of the best action to perform. In the first case, we suppose that there exists a computation S_j which affects the agent's estimation of the utility of the alternative action β so that

$$E[V(S_j)] = \int_{\hat{U}_I([\alpha])}^{\infty} p_{\beta,j}(x)(x - \hat{U}_I^S([\alpha]))dx - TC(S_j), \tag{4}$$

where $p_{\beta,j}$ is the probability density function for $\hat{U}_I^{S.S_j}([\beta])$. In the second case, there exists a computation S_k which affects the agent's estimation of the utility of the default action α so that

$$E[V(S_k)] = \int_{-\infty}^{\hat{U}_I([\beta])} p_{\alpha,k}(x)(\hat{U}_I^{S}([\beta]) - x)dx - TC(S_k), \tag{5}$$

where $p_{\alpha,k}$ is the probability density function for $\hat{U}_I^{S.S_k}([\alpha])$. If there are n computational actions and m base-level actions, then each meta-level reasoning step will require computing each of Equations 4 and 5 nm times. If the distributions governing utility estimates are simple in form (*e.g.,* normal distributions), computing the integrals in Equations 4 and 5 can be done quite efficiently.

There are a number of assumptions that Russell and Wefald make in their analysis. First, it is assumed that the agent considers only single computation steps, estimates their ultimate effect, and then chooses the step appearing to have highest benefit. This is referred to as the *meta-greedy* assumption. Second, it is assumed that the agent will act as though it will take at most one more search step. This is referred to as the *single-step* assumption. Finally, it is assumed that a computational action will change the expected utility estimate for exactly one base-level action. In Russell and Wefald's state-space search paradigm, this is referred to as the *subtree-independence* assumption.

The assumptions stated in the previous paragraph may seem overly restrictive, but it is quite difficult to avoid these or similar assumptions in general. Pearl [40] identifies two assumptions that most practical meta-reasoning systems ascribe to: *no competition*, each information source is evaluated in isolation from all the others, and *one-step horizon*, we consult at most one additional information source before committing to a base-level action. Assessments of information sources based on the no-competition and one-step-horizon assumptions are referred to as *myopic*, and most practical systems employ myopic decision policies [3, 12].

The next piece of research that we consider in this section is due to Etzioni [19], and it borrows from the Russell and Wefald work, and builds on the early work of Simon and Kadane [46] on satisficing search. It is particularly interesting for the fact that it attempts to combine the sort of goal-driven behavior prevalent in artificial intelligence with the decision theoretic view of maximizing expected utility.[5] In Etzioni's model, the agent is given a set

[5]See Dean and Wellman [15] for a discussion of the differences between the artificial intelligence and

of goals $\{G_1, G_2, \ldots, G_n\}$, a set of methods M, and a deadline B. The agent attempts to determine a sequence of methods

$$\sigma = m_{1,1}, m_{2,1}, \ldots, m_{k_1,1}, m_{1,2}, m_{2,2}, \ldots, m_{k_2,2}, \ldots, m_{1,n}, m_{2,n}, \ldots, m_{k_n,n}$$

where $m_{i,j}$ is the ith method to be applied to solving the jth goal. The idea is that the agent will apply each method in turn until it either runs out of methods or achieves the goal, at which point it will turn its attention to the next goal. The expected utility of σ is

$$\begin{aligned}
&E[\hat{U}(\sigma)] = \\
&\quad E[\hat{U}(m_{1,1})] + E[\hat{U}(m_{2,1})](1 - \Pr(m_{1,1})) + \ldots + E[\hat{U}(m_{k_1,1})] \prod_{i=1}^{k_1-1} (1 - \Pr(m_{i,1})) + \\
&\quad E[\hat{U}(m_{1,2})] + E[\hat{U}(m_{2,2})](1 - \Pr(m_{1,2})) + \ldots + E[\hat{U}(m_{k_2,2})] \prod_{i=1}^{k_2-1} (1 - \Pr(m_{i,2})) + \\
&\quad \ldots \\
&\quad E[\hat{U}(m_{1,n})] + E[\hat{U}(m_{2,n})](1 - \Pr(m_{1,n})) + \ldots + E[\hat{U}(m_{k_n,n})] \prod_{i=1}^{k_n-1} (1 - \Pr(m_{i,n})).
\end{aligned}$$

In this simple model, no provision is made for switching back and forth between goals, and, except for ignoring the remaining methods for a given goal once that goal has been achieved, no provision is made for modifying the search as new information becomes available. Etzioni defines the expected *opportunity cost* (γ_B) of a method m for a deadline B as

$$E[\gamma_B(m)] = E[\hat{U}(\sigma^*_B)] - E[\hat{U}(\sigma^*_{B-TC(m)})],$$

where σ^*_B is the optimal method sequence for a deadline B, and $TC(m)$ is the expected time to carry out method m. In addition, the expected *gain* (G_B) of a method m for a deadline B is defined to be

$$E[G_B(m)] = E[\hat{U}(m)] + E[\gamma_B(m)].$$

He then shows that by repeatedly choosing the method whose expected gain is maximal an agent will construct an optimal method sequence.

From one point of view, Etzioni's work is not about meta-reasoning at all; his work is concerned with ordinary sequential decision problems. For these problems, Etzioni points

decision theoretic perspectives.

out that, in certain cases, the cost of determining an optimal method sequence can be quite high. In other words, we can't ignore the cost of meta-level reasoning in the decision-making model. His analysis showing that sorting methods on their marginal utility can often result in optimal or near-optimal method sequences is exactly the sort of analysis required to justify a particular meta-level reasoning.

In the case in which the agent has a single goal and multiple methods for achieving it, the requisite meta-reasoning is easy. In particular, suppose that there is a constant opportunity cost γ per unit of time spent on the goal, and for each method $m \in M$ the agent has an expected time cost $E[TC(m)]$, an expected utility estimate $E[\hat{U}(m)]$, and a probability $\Pr(m)$ of achieving the goal using that method. The expected gain of a method m is just

$$E[G(m)] = E[\hat{U}(m)] - \gamma E[TC(m)],$$

and the task is to find σ so as to maximize

$$E[\hat{U}(\sigma)] = \sum_{i \text{ s.t. } E[G(m_i)]>0} E[G(m_i)] \prod_{k=0}^{i-1} (1 - \Pr(m_k)).$$

Etzioni claims, and it is easy to verify, that, by sorting the methods in increasing order using $\frac{E[G(m)]}{\Pr(m)}$ as a key, an agent can construct an optimal method ordering.

In the above case, it is plausible to assume that the cost of meta-reasoning (*i.e.*, the time spent calculating $\frac{E[G(m)]}{\Pr(m)}$ for each method m and sorting using the results) is negligible in comparison with the cost of applying a method. In the case of an agent faced with multiple goals even where there is only one method for each goal, it is more difficult to make such an assumption. By reducing the *knapsack problem* [21] to the problem of computing the expected opportunity cost, Etzioni shows that computing the expected opportunity cost of a method is NP-complete. It is not too surprising that there are some hard problems lurking among the deliberation scheduling problems that underlie decision theoretic control of inference. It should be pointed out, however, that all we should really be concerned with is the expected cost of meta-reasoning, and that, in many practical applications, approximations are much preferred to even polynomial-time methods for computing exact solutions. Etzioni suggests using Garey and Johnson's factor-of-two approximation [21], but it should be noted that the knapsack problem is a number problem for which there exist pseudo-polynomial time algorithms and good branch-and-bound approximations. These branch-and-bound algorithms have exponential-time worst-case behavior, but their expected performance is

such that many practitioners consider knapsack tractable. In the next section, we will see how approximation algorithms for computationally expensive problems can provide us with even greater flexibility in allocating processor time to decision procedures.

While Etzioni's invocation of asymptotic complexity as a measure of difficulty may not be particularly appropriate in this case, it does force the reader to reconsider the assumptions regarding the time cost of meta-reasoning. For instance, if n is small and the average time cost of the methods in M is high, then it may even be reasonable to perform a meta-computation whose worst-case behavior is exponential in n. It may even be useful to add another level of meta-reasoning to reason about various alternative scheduling algorithms.

Just what is the structure of the decision making process that we are seeking to control? In the Russell and Wefald model, object-level decision making involves fixed-duration computations that attempt to provide a better assessment of a single base-level action. In the next section, we consider problems in which the meta-level reasoner sacrifices some of its control over object-level decision making in order to simplify meta-level decision making. In particular, we consider decision procedures that return estimates that improve with additional allocations of processor time. The ability to preempt decision procedures at any time during their computation simplifies deliberation scheduling in many cases.

5 Flexible Computations and Anytime Algorithms

In this section, we consider two independently developed but closely related approaches to improving the decision theoretic control of problem solving. The two approaches are due to Dean and Boddy [13] and Horvitz [28]. Horvitz refers to his decision procedures as *flexible computations* and Dean and Boddy refer to theirs as *anytime algorithms*, but the basic idea behind the two proposals is the same, and we will use the two terms interchangeably.

In the ideal flexible computation, the object-related value of the result returned by a decision procedure is a continuous function of the time spent in computation. The notion of "object-related value" of a result is to be contrasted with the "comprehensive value" of a system's response to a given state; the latter refers to the overall utility of the response and the former is some measure of the value of the result apart from its use in a particular set of circumstances. Object-related value is exactly Russell and Wefald's intrinsic utility. In

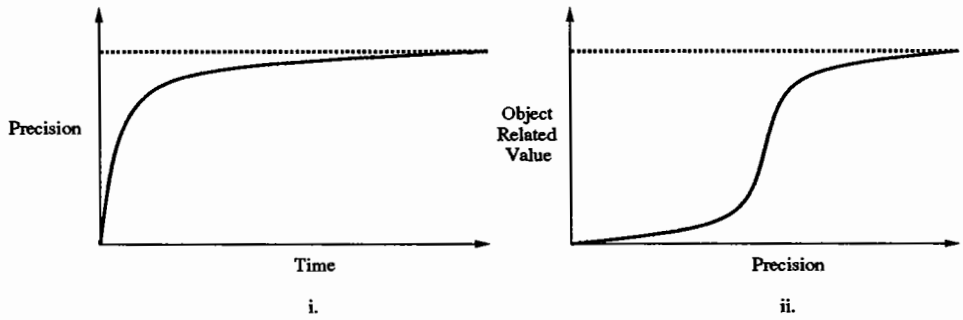

Figure 1: Plots relating (i) time spent in computation to the precision of a probabilistic calculation, and (ii) precision to the value associated with getting the diagnosis correct and treating the patient accordingly (after [28]).

some cases, the task of relating a result to the comprehensive value of the overall response can be quite complex. This is especially so in cases in which there are several results from several different decision procedures. In these cases, it is often convenient to make the assumption that the value function is *separable* so that the comprehensive value of the system's response can be computed as the sum of the value of a sequence of outcomes.

We assume that a flexible computation can be interrupted at any point during computation—hence, the name "anytime"—to return an answer whose object-related value increases as it is allocated additional time. Horvitz provides a good example of a flexible computation and an analysis of its object-related value drawn from the health care domain. Suppose that we have an anytime algorithm that computes a posterior distribution for a set of possible diagnoses given certain information regarding a particular patient. Figure 1.i (from [28]) shows a graph that relates the precision of the result returned by this algorithm to the time spent in computation. The object-related value can be determined as a function of precision by considering the expected utility of administering a treatment based on a diagnosis of a given precision ignoring when the treatment is administered (see Figure 1.ii).

The comprehensive value of computation is meant to account for the costs and benefits related to the time at which the results of decision procedures are made use of to initiate physical actions. Figure 2.i (from [28]) indicates how a physician might discount the object-related value of a computation as a function of delay in administering treatment. The comprehensive value of computation is shown in Figure 2.ii and is obtained by combining

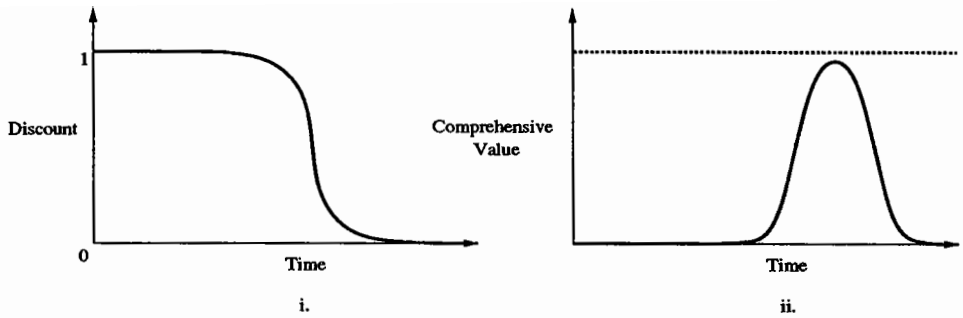

Figure 2: Plots indicating (i) a discounting factor for delayed treatment, and (ii) the comprehensive value of computation as a function of time (after [28]).

the information in Figures 1.ii and 2.i. This method of combining information assumes both time-cost separability and one-step horizon.

Both Horvitz and Dean and Boddy note that the most useful sort of flexible computations are those whose object-related value increases monotonically over some range of computation times. Dean and Boddy [13] employ decision procedures that are monotonic throughout the range of computation times, and exploit this fact to expedite deliberation scheduling for a special class of planning problems referred to as *time-dependent planning problems*. A planning problem is said to be time-dependent if the time available for responding to a given event varies from situation to situation. In their model, a predictive component, whose time cost is not considered, predicts events and their time of occurrence on the basis of observations, and the planning system is given the task of formulating a response to each event and executing it in the time available before the event occurs. This model generalizes on the multiple-goals/single-method-for-each model of Etzioni described in the previous section, by allowing each goal to have a separate deadline. If the responses to the different events are independent, then the task of deliberation scheduling can be stated in terms of maximizing

$$\mu = \sum_{e \in E} \mathsf{value}(\mathsf{response}(e)),$$

where E is the set of all events predicted thus far. It is assumed that there is exactly one decision procedure for each type of event likely to be encountered, and that there are statistics on the performance of these decision procedures. The statistics are summarized

in what are called *performance profiles* which are essentially the same as the graphs used by Horvitz in his analysis (*e.g.*, see Figure 1.ii). Dean and Boddy provide an $O(|E|)$ algorithm for deliberation scheduling involving time-dependent planning problems for a wide class of anytime decision procedures. The algorithm uses a simple greedy strategy working backward from the last predicted event, choosing the decision procedure whose expected gain computed from the performance profiles is greatest [13].

It is worth considering why the NP-completeness result reported by Etzioni does not apply in this more general case. In job-shop scheduling, if it is possible to suspend, and later resume, a job, then many otherwise difficult problems become trivial [24, 5]. Such (preemptive) scheduling problems are somewhat rare in real job shops given that there is often significant overhead involved is suspending and resuming jobs (*e.g.*, traveling between workstations or changing tools), but they are considerably more common with regard to purely computational tasks (*e.g.*, suspending and resuming Unix processes). In many scheduling problems, each job has a fixed cost and requires a fixed amount of time to perform; spending any less than the full amount yields no advantage. This is the case in the decision procedures considered by Etzioni. If, however, the decision procedures for computing appropriate actions are preemptible and provide better answers depending upon the time available for deliberation, then the task of deliberation scheduling is considerably simplified. Anytime decision procedures thus provide more flexibility in responding to time-critical situations, and simplify the task of allocating processor time in cases where there is contention among several decision tasks.

For the multiple-goals/single-method-for-each problem described in the previous section, Etzioni suggests using a factor-of-two approximation to avoid potential combinatorics in deliberation scheduling. Rather than always simply applying the factor-of-two approximation, we can design an anytime approximation algorithm and allocate it some amount of processor time based on expectations regarding its performance. The fully-polynomial approximation scheme[6] of Ibarra and Kim [33] for solving the optimization version of the knapsack problem serves nicely as the basis for an anytime approximation algorithm for choosing method sequences. The simplest approach would be to classify the base-level

[6]See Garey and Johnson [21] for a discussion of fully-polynomial approximation schemes for NP-complete problems. For our purposes, an approximation scheme S for a problem Π takes an instance I_Π and a precision requirement $\epsilon > 0$ and returns a candidate solution that is within ϵ of the optimal solution. Such a scheme is said to be fully polynomial just in case the time complexity of S is bounded by a polynomial function of $\frac{1}{\epsilon}$ and the size of I_Π.

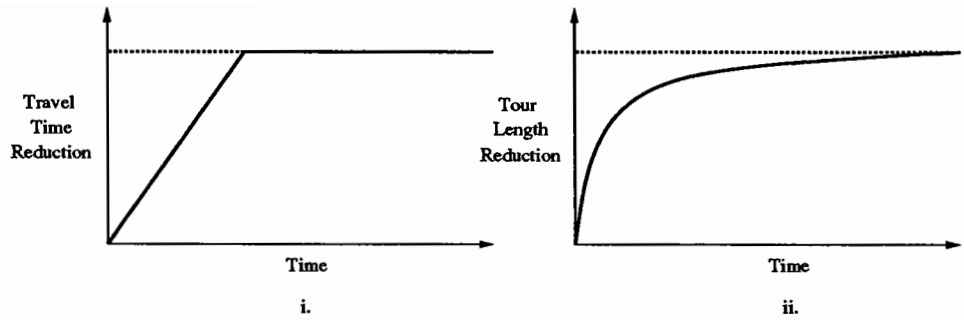

Figure 3: Performance profiles relating (i) the expected savings in travel time to time spent in path planning, and (ii) the expected reduction in the length of a returned tour as a function of time spent in tour improvement.

problems in terms of, say, the number of goals and the length of time until the deadline, and gather statistics on the utility derived from invoking the approximation scheme with different precision requirements. Whether or not this more complicated approach to deliberation scheduling performs better than the factor-of-two approximation will depend upon the specifics of the application and how efficiently the algorithms are realized. It is easy to imagine applications, however, for which the expected performance of the system will be improved by using such a scheme.

The use of flexible computations can also simplify problems in which one decision procedure produces an intermediate result that is used as input to a second decision procedure. Boddy and Dean [4] investigate one such problem involving a robot courier assigned the task of delivering packages to a set of locations. The robot has to determine both the order in which to visit the locations, referred to as a *tour*, and, given a tour, plan paths to traverse in moving between consecutive locations in the tour. To simplify the analysis, it is assumed that the robot's only concern is time; it seeks to minimize the total amount of time consumed both in sitting idly deliberating about what to do next, and in actually moving about the environment on its errands. Furthermore, it is assumed that there is no advantage to the robot in starting off in some direction until it knows the first location to be visited on its tour. Finally, while the robot can deliberate about any subsequent paths while traversing a path between consecutive locations in the tour, it must complete the planning for a given path before starting to traverse that path.

The two primary components of the decision making process involve generating the tour and planning the paths between consecutive locations in the tour. The first is referred to as *tour improvement* and the second as *path planning*. Boddy and Dean employ iterative refinement approximation routines for solving each of these problems, and gather statistics on their performance to be used at run-time in guiding deliberation scheduling. The statistics are summarized in what are called *performance profiles*. Figure 3 (from [4]) shows the profiles for path planning and tour improvement. Figure 3.i shows how the expected savings in travel time increase as a function of time spent in path planning. Figure 3.ii shows how the expected length of the tour decreases as a fraction of the shortest tour for a given amount of time spent in tour improvement. In the analysis described in [4], this performance estimate is independent of the initially selected tour. We assume that the robot starts out with an initial randomly selected tour. Given the length of some initial tour, the expected reduction in length as a function of time spent in tour improvement, and some assumptions about the performance of path planning, the robot can figure out exactly how much time to devote to tour improvement in order to minimize the time spent in stationary deliberation and combined deliberation and traversal.

There currently is no general theory about how to combine anytime algorithms, and neither is there likely to be in the near future. For cases in which the decision problems are not independent, there is not a great deal that we can say. However, for the case of independent decision problems for which anytime decision procedures exist, or for which a pipelined sequence of anytime decision procedures exist, as in the robot courier problem, there is a great deal of interesting research to be done; research that can draw heavily on the scheduling and combinatorial optimization literature.

It is worth pointing out some connections between the Russell and Wefald work and that of Dean and Boddy and Horvitz. The Russell and Wefald work can be seen as trying to construct an optimal anytime algorithm; a single algorithm that operates by calculating a situation-specific estimate of utility using only local information, just as subscribed by information value theory. It should be possible to apply the Russell and Wefald approach to scheduling anytime algorithms. For some purposes (*e.g.*, the game-playing and search applications described in [44]), the monolithic approach of Russell and Wefald seems perfectly suited. For other applications (*e.g.*, the robotic applications described in [4] or the intensive-care applications described in [29]), it is quite convenient to think in terms of scheduling existing approximation algorithms.

Deliberation scheduling can be seen as a means of building in knowledge about how to improve run-time performance. There are occasions, however, in which the time required to apply that knowledge is not available at run time, and it becomes reasonable to make certain choices concerning the allocation of computational resources at design time. The next section considers design-time tradeoffs for improving system performance.

6 Compiling Problem Solving Systems

In the previous sections, we were concerned with the design of systems that, given expectations about the performance of decision-making routines, were able to make appropriate tradeoffs at run-time so as to maximize expected utility. Another approach to building systems capable of good performance in time-critical situations involves making certain inferences at design time and caching those inferences for use at run time in order to improve the system's response time. Other researchers have suggested compiling domain models to guarantee bounded response time [34, 42]. Generally, the result of compilation is a table or circuit whose space requirements are an important factor in assessing the value of a given compilation method. Usually, the object is to improve response time without sacrificing decision quality; when this cannot be done (*e.g.*, the storage requirements for caching are substantial) it becomes necessary to consider tradeoffs. The approaches described in this section are noteworthy for their use of a decision theoretic criterion for trading space for response time.

Heckerman, Breese, and Horvitz [27] investigate a simple form of tradeoff that involves improving response time by compiling decision models. In their model, the utility of a state depends on whether or not a particular hypothesis H is true and whether or not an action D is taken. We will assume that, if H is true, the action D should be taken, and otherwise the action $\neg D$ is appropriate. We can define a threshold probability of H, call it p^*, such that the agent is indifferent about acting one way or the other:

$$p^*U(H, D) + (1 - p^*)U(\neg H, D) = p^*U(H, \neg D) + (1 - p^*)U(\neg H, \neg D).$$

The agent is not able to observe H directly, and, hence, must infer whether or not H is true on the basis of certain pieces of observed evidence, $E_1, E_2, \ldots, E_n$. Thus the agent should perform the action D if and only if $\Pr(H|E_1, E_2, \ldots, E_n) > p^*$. The resulting decision model

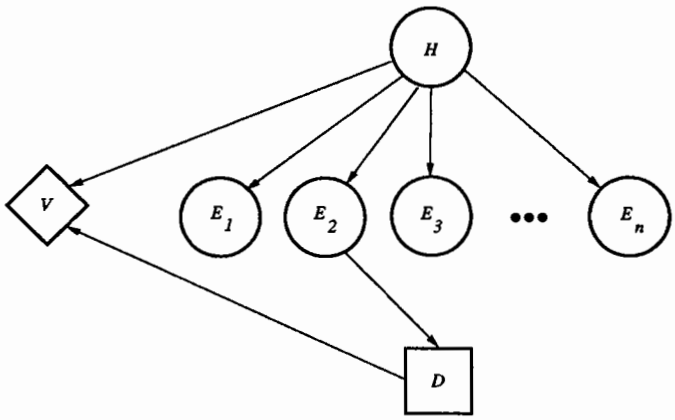

Figure 4: Decision model for the diagnosis problem (after [27])

(depicted graphically in Figure 4) (from [27]) is represented as an *influence diagram* [32] that captures the causal and informational dependencies between chance variables (indicated as circles) and between chance variables and decision variables (indicated as boxes), and the value of states of the world corresponding to particular instantiations of the chance and decision variables (indicated as diamonds).

Heckerman, Breese, and Horvitz reformulate the decision problem in terms of log-likelihood ratios, and, by making certain independence assumptions, they reduce the decision problem to computing

$$W = \sum_{i=1}^{n} w_i$$

where the w_i are the weights accorded to the E_i. The agent should perform the action D if and only if $W > W^*$ where W^* is the log-likelihood equivalent of p^*. We will refer to the strategy of computing the weights at run time as the *compute* strategy.

As an alternative to computing the sum of the weights of evidence at run time, the agent might consider all possible combinations of evidence and compile a table indicating whether or not to act for each possible combination. If memory is inexpensive and response time critical, then this might be an attractive alternative. In general, however, it will be prohibitively expensive to compile a table for all possible combinations of the evidence, and, hence, if the agent wants to speed its response time by compiling a table, it will have to limit its attention to a subset of the evidence. Suppose that the agent chooses m pieces of

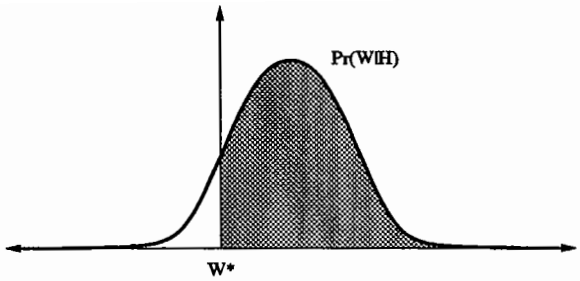

Figure 5: The probability that the total evidential weight will exceed the threshold is determined by summing the area under the curve for the distribution of W given H and above the threshold weight W_m (after [27]).

evidence, $\{E_{c_1}, E_{c_2}, \ldots, E_{c_m}\} \subseteq \{E_1, E_2, \ldots, E_n\}$, to use in compiling a table of responses. For each of the 2^m possible combinations of the m variables, we compute

$$W_m = \sum_{i=1}^{m} w_{c_i}$$

at compile time, and store D in the table if $W_m > W^*$ and $\neg D$ otherwise. At run time, the agent simply uses the evidence as an index to lookup the appropriate entry in the table. We will refer to a strategy of compiling a table for m pieces of evidence as a *compile* strategy.

Note that the advantage of the compute strategy is that it takes all of the evidence into account; the disadvantage is that there may be some delay between the time that the evidence is observed and the time that the agent responds to the evidence. The compile strategy may enable the agent to respond more quickly, but at the cost of ignoring some of the evidence and providing storage for a decision table whose size is exponential in the number of pieces of evidence accounted for in the reduced model.

In the following, let EV_{compute} indicate the expected value of the agent using the compute strategy for a single instance of the decision problem, PC^{H}_{compute} and $PC^{\neg H}_{\text{compute}}$ indicate the cost due to computing delays in the case in which H is true and the case in which H is not, and MC_{compute} indicate the one time cost of memory for the compute strategy. Assume similar quantities for the compile strategy. In order to compare the compute strategy against different compile strategies (*i.e.*, compilation involving different subsets of $\{E_1, E_2, \ldots, E_n\}$), Heckerman, *et al.* introduce formulae for determining the *net inferential*

value of a given strategy.

$$NIV_{\text{compute}} = \rho[EV_{\text{compute}} - \Pr(H)PC^{H}_{\text{compute}} - \Pr(\neg H)PC^{\neg H}_{\text{compute}}] - MC_{\text{compute}}$$

$$NIV_{\text{compile}^m} = \rho[EV_{\text{compile}^m} - \Pr(H)PC^{H}_{\text{compile}^m} - \Pr(\neg H)PC^{\neg H}_{\text{compile}^m}] - MC_{\text{compile}^m}$$

where the NIV_{compile^m} depends upon the particular choice of evidence, and ρ is a factor "that converts the expected value of each policy on a single instance to a summary (present) value for a series of problem instances over the life of the system." Given the above, the agent designer should choose the compute strategy over the compile strategy if and only if

$$NIV_{\text{compute}} > NIV_{\text{compile}^m}.$$

In the analysis presented in [27], $PC^{\neg H}_{\text{compute}}$, PC^{H}_{compute}, and MC_{compute} are linear functions of n, the total number of evidence variables in the complete model, $PC^{\neg H}_{\text{compile}^m}$ and $PC^{H}_{\text{compile}^m}$ are linear functions of m, the total number of evidence variables in the restricted compilation model, and MC_{compile^m} is a linear function of 2^m. The formulae for the expected value of using the compute and compile strategies for a single instance of the decision problem are given as follows:

$$\begin{aligned} EV_{\text{compute}} &= [\Pr(W > W^*|H)U(H,D) + \Pr(W \leq W^*|H)U(H,\neg D)]\Pr(H) + \\ &\quad [\Pr(W > W^*|\neg H)U(\neg H,D) + \Pr(W \leq W^*|\neg H)U(\neg H,\neg D)]\Pr(\neg H) \end{aligned}$$

$$\begin{aligned} EV_{\text{compile}^m} &= [\Pr(W_m > W^*|H)U(H,D) + \Pr(W_m \leq W^*|H)U(H,\neg D)]\Pr(H) + \\ &\quad [\Pr(W_m > W^*|\neg H)U(\neg H,D) + \Pr(W_m \leq W^*|\neg H)U(\neg H,\neg D)]\Pr(\neg H) \end{aligned}$$

The only trick to using the above to decide whether to use the compute or compile strategy is determining the probabilities involving the weights (*e.g.*, $\Pr(W > W^*|H)$). Assuming that n is large (as it should be for us to take seriously the cost of computing W), then we can compute the first two moments for the each of the weights given H and combine them to approximate the distribution of W given H using the central limit theorem. Using the resulting approximations for $\Pr(W_m|H)$, $\Pr(W_m|\neg H)$, $\Pr(W|H)$, and $\Pr(W|\neg H)$, we can determine the values for the terms needed to compute EV_{compute} and EV_{compile} (see Figure 5).

Heckerman, *et al.* go on to consider relaxing certain assumptions (specifically, allowing multiple-valued hypothesis, evidence, and decision variables and introducing alternatives

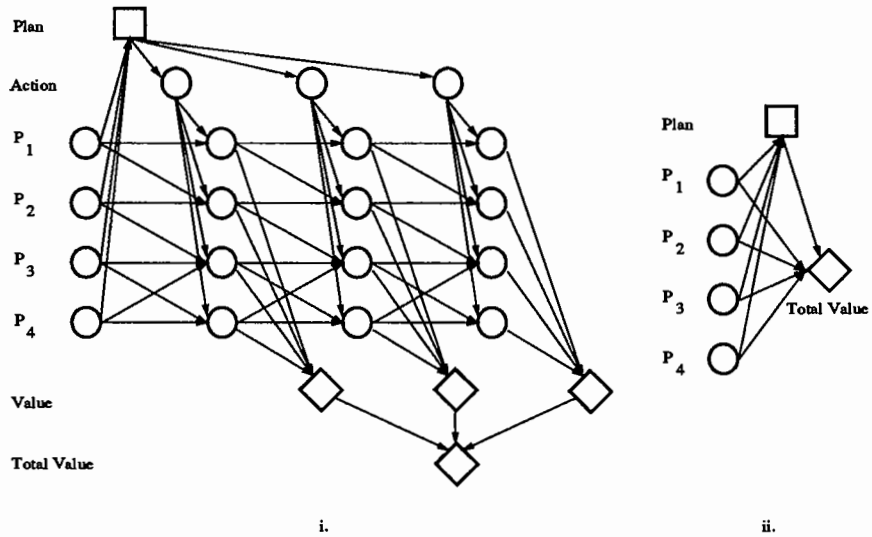

Figure 6: Two influence diagrams indicating (i) a complete decision model for reasoning about plans, and (ii) a reduced version of the decision model obtained by absorbing chance nodes in the complete model.

to caching complete tables, in the form of caching situation/action rules in asymmetrical trees), and methods for considering what subsets of the set of all evidence variables to consider for compilation. What they don't consider, and what might be worth pursuing, are mixed strategies involving some amount of design-time compilation and some amount of run-time inference.

If the basic methods described by Heckerman, *et al.* for evaluating the expected performance of decision models used for time-critical applications turn out to be practical for realistic decision problems, then we will want to try out more sophisticated models for reasoning about plans and change over time. Kanazawa and Dean [35] describe a model for reasoning about time, causation, and action that can be cast as an influence diagram. Given a set $\mathcal{P}$ of propositions and a set $\mathcal{T}$ of time points, we can define a set of chance variables from $\mathcal{P} \times \mathcal{T}$ representing the truth of various propositions at different points in time. By quantifying the dependencies between these chance variables, we can specify a model of change over time referred to as a *temporal Bayes net* [14].

The model described in [35] generalizes on this basic model of change over time to include

actions so as to provide a decision model for selecting plans. Figure 6.i shows an example of such a model depicted as an influence diagram. Each row, except those corresponding to decision variables or value functions, indicates a proposition or quantity that changes over time, and each column indicates a different point in time. Kanazawa and Dean consider possible tradeoffs involved in improving the performance of reasoning systems using such a model for decision making. In particular, they consider trading accuracy for time by employing approximation schemes for evaluating probabilistic models [8, 29]. They also consider trading space for time by eliminating chance variables in the decision model using a method of conditioning called *node absorption* [45]. By eliminating chance variables at design time, it is possible to dramatically improve the time required to evaluate the model. Such improvements occur for both exact and approximate evaluation techniques. Figure 6.ii shows a version of the model shown in Figure 6.i obtained by repeated use of node absorption.

In general, node absorption can result in an increase in the space required to store the model; there will be fewer nodes in the resulting graph, but the space required to store the conditional probabilities quantifying the dependencies may increase significantly. However, given the structure of temporal Bayes nets, the net increase in space is generally acceptable and more than offset by the resulting reduction in evaluation time. It would be interesting to extend the techniques of Heckerman, *et al.* to evaluate at design time various alternative approximation schemes and methods of simplifying the decision model. The biggest barriers to making such extensions practical will likely be due to the combinatorics of action selection and the difficulties involved in obtaining an accurate model of the environment in the first place.

7 Conclusions and Directions for Future Research

This paper provides only a sketch of current work on problem solving methods for time-critical applications. There is a great deal of excellent research that we did not cover, simply because it did not fit into the structure of the presentation. In particular, we did not say anything significant about architectures for real-time control [1, 7], or relate how the search community is beginning to address real-time issues [25, 36, 47]. Regarding search, Hansson and Mayer's work [25] predicts that we will find many of the standard techniques in heuristic search as emergent properties of mechanisms that employ Bayesian inference

and decision-theoretic control of inference. All of this work is serving to shape a new field of research.

The next few years will see a marked increase in the effort directed at time-critical problem solving and resource-limited reasoning. We need to extend the current approaches to handle computational models that reflect the complexity of existing problem-solving systems. For instance, how might an agent deal with multiple tasks, perhaps deciding to act with regard to one task while continuing to deliberate about others. We need experience with real applications so that the research will be driven by real issues and not artifacts of our mathematical models. We need to reconcile the goal-oriented, resource-bounded perspective of artificial intelligence with the idealized, optimizing perspective taken in the decision sciences.

This paper makes use of Howard's information value theory as a basis from which to start in analyzing systems with limited computational resources. All of the approaches surveyed in this paper can be seen as extensions of the basic idea of assessing the value of information sources. The approaches surveyed here depart from information value theory when they attempt to account for the cost of inference, including the computational cost of assessing the value of information sources. It would seem that the theory of experimental design [20, 38] which is concerned with the problem of maximizing the information gained from performing experiments under cost constraints might provide a source of additional techniques that could be applied in controlling inference for time-critical applications.

All of the approaches surveyed in this paper made rather restrictive assumptions in order to avoid the combinatorics involved in dealing with unlimited decision-making horizons and complicated interactions between information sources. For practical problems, it is unlikely that we will be able to entirely relax the one-step horizon and no-competition assumptions that characterize myopic decision policies. An interesting area for future research involves identifying and dealing with restricted types of interactions and providing a disciplined approach to extending decision-making horizons. It would also be useful to explore methods of extending the anytime algorithm approaches to handle more situation-specific information.

The research on compiling decision models is just beginning, and one area that appears particularly interesting to investigate involves mixed strategies for combining design-time compilation and run-time inference. Another area that was not covered in this survey, but is of considerable interest involves learning control knowledge in the form of statistics to

support decision-theoretic control of inference. Two of the papers covered in this survey [44, 19] describe interesting techniques that address learning issues.

The work in time-critical problem solving will have far reaching implications for the whole research community. Time is, after all, an issue in any problem solving task. Theoretical results concerning agents with limited computational resources should shed light on a number of basic representation issues. For instance, the notion of a "plan" as a persistent belief does not make sense until you take computational considerations into account. Plans enable a system to amortize the cost of deliberation over an interval of time. If time were not an issue, there would be no justification in committing to a plan. What are the tradeoffs involved in generating a partial plan? What are the costs and benefits of compiling a detailed plan to use in a situation in which there will be very little time for computing appropriate responses. These are just a few of the questions that can be addressed once we begin to account for the time spent in problem solving.

References

[1] Agogino, A. M., Srinivas, S., and Schneider, K., Multiple Sensor Expert System for Diagnostic Reasoning, Monitoring, and Control of Mechanical Systems, *Mechanical Systems and Signal Processing*, (1988).

[2] Barnett, V., *Comparative Statistical Inference*, (John Wiley and Sons, New York, 1982).

[3] Ben-Basat, M., Myopic Policies in Sequential Classification, *IEEE Transactions on Computers*, **27** (1978) 170–174.

[4] Boddy, Mark and Dean, Thomas, Solving Time-Dependent Planning Problems, *Proceedings IJCAI 11, Detroit, Michigan*, IJCAI, 1989, 979–984.

[5] Bodin, L. and Golden, B., Classification in Vehicle Routing and Scheduling, *Networks*, **11** (1981) 97–108.

[6] Brachman, Ronald J., Levesque, Hector J., and Reiter, Raymond, (Eds.), *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, (Morgan-Kaufman, Los Altos, California, 1989).

[7] Breese, John S. and Fehling, Michael R., Decision-Theoretic Control of Problem Solving: Principles and Architecture, *Proceedings of the 1988 Workshop on Uncertainty in Artificial Intelligence, Minneapolis, MN*, 1988, 30–37.

[8] Chavez, R. Martin, *Fully Polynomial Randomized Approximation Schemes for the Bayesian Infererencing Problem*, Report KSL-88-72, Section on Medical Informatics, Stanford University School of Medicine, 1988.

[9] Chernoff, Herman and Moses, Lincoln E., *Elementary Decision Theory*, (John Wiley and Sons, New York, 1959).

[10] Davis, Randall, Teiresias: Applications of Meta-Level Knowledge, Davis, Randall and Lenat, Douglas B., (Eds.), *Knowledge-Based Systems in Artificial Intelligence*, (McGraw-Hill International Book Company, 1982), 227–490.

[11] de Kleer, Johan, Doyle, Jon, Steele Jr., Guy L., and Sussman, Gerald Jay, AMORD: Explicit Control of Reasoning, Brachman, Ronald J. and Levesque, Hector J., (Eds.), *Readings in Knowledge Representation*, (Morgan Kaufmann, Los Altos, CA, 1985), chapter 19, 346–355. Originally published in 1977.

[12] de Kleer, Johan and Williams, Brian C., Diagnosing Multiple Faults, *Artificial Intelligence*, **32**(1) (1987) 97–130.

[13] Dean, Thomas and Boddy, Mark, An Analysis of Time-Dependent Planning, *Proceedings AAAI-88, St. Paul, Minnesota*, AAAI, 1988, 49–54.

[14] Dean, Thomas and Kanazawa, Keiji, A Model for Reasoning About Persistence and Causation, *Computational Intelligence*, **5**(3) (1989) 142–150.

[15] Dean, Thomas and Wellman, Michael, On the Value of Goals, Tenenberg, Josh, Weber, Jay, and Allen, James, (Eds.), *Proceedings from the Rochester Planning Workshop: From Formal Systems to Practical Systems*, 1989, 129–140.

[16] Doyle, Jon, *A Model for Deliberation, Action, and Introspection*, Technical Report AI-TR-581, MIT AI Laboratory, 1980.

[17] Doyle, Jon, Reasoning, Representation, and Rational Self-Government, Ras, Zbigniew W., (Ed.), *Methodologies for Intelligent Systems, 4*, New York, North-Holland, 1989, 367–380.

[18] Edwards, Ward, Dynamic Decision Theory and Probabilistic Information Processing, *Human Factors*, 4 (1962) 59–73.

[19] Etzioni, Oren, Tractable Decision-Analytic Control, In Brachman et al. [6], 114–125.

[20] Fedorov, V., *Theory of Optimal Experimental Design*, (Academic Press, New York, 1972).

[21] Garey, Michael R. and Johnson, David S., *Computing and Intractibility: A Guide to the Theory of NP-Completeness*, (W. H. Freeman and Company, New York, 1979).

[22] Genesereth, Michael R., An Overview of Metalevel Architecture, *Proceedings AAAI-83, Washington, D.C.*, AAAI, 1983, 119–123.

[23] Good, I.J., A Five Year Plan for Automatic Chess, *Machine Intelligence*, 4 (1962) 59–73.

[24] Graham, R.L., Lawler, E.L., Lenstra, J.K., and Rinnooy Kan, A.H.G., Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey, *Proceedings Discrete Optimization, Vancouver*, 1977.

[25] Hansson, Othar and Mayer, Andrew, The Optimality of Satisficing Solutions, *Proceedings of the 1988 Workshop on Uncertainty in Artificial Intelligence, Minneapolis, MN*, 1988, 148–157.

[26] Hayes-Roth, Barbara, Washington, Richard, Hewett, Rattikorn, Hewett, Michael, and Seiver, Adam, Intelligent Monitoring and Control, *Proceedings IJCAI 11, Detroit, Michigan*, IJCAI, 1989, 243–249.

[27] Heckerman, David E., Breese, John S., and Horvitz, Eric J., The Compilation of Decision Models, *UW89, Windsor, Ontario*, 1989, 162–173.

[28] Horvitz, Eric J., Reasoning About Beliefs and Actions Under Computational Resource Constraints, *Proceedings of the 1987 Workshop on Uncertainty in Artificial Intelligence, Seattle, Washington*, 1987.

[29] Horvitz, Eric J., Suermondt, H. Jacques, and Cooper, Gregory F., *Bounded Conditioning: Flexible Inference for Decisions Under Scarce Resources*, Technical Report KSL-89-42, Stanford Knowledge Systems Laboratory, 1989.

[30] Howard, Ronald A., Information Value Theory, *IEEE Transactions on Systems Science and Cybernetics*, **2**(1) (1966) 22–26.

[31] Howard, Ronald A., Value of Information Lotteries, *IEEE Transactions on Systems Science and Cybernetics*, **3**(1) (1967) 54–60.

[32] Howard, Ronald A. and Matheson, James E., Influence Diagrams, Howard, Ronald A. and Matheson, James E., (Eds.), *The Principles and Applications of Decision Analysis*, (Strategic Decisions Group, Menlo Park, CA 94025, 1984).

[33] Ibarra, O. H. and Kim, C.. E., Fast Approximation Algorithms for the Knapsack and Sum of the Subset Problems, *Journal of the ACM*, **22** (1975) 463–468.

[34] Kaelbling, Leslie Pack, Goals as Parallel Program Specifications, *Proceedings AAAI-88, St. Paul, Minnesota*, AAAI, 1988, 60–65.

[35] Kanazawa, Keiji and Dean, Thomas, A Model for Projection and Action, *Proceedings IJCAI 11, Detroit, Michigan*, IJCAI, 1989, 985–990.

[36] Korf, Richard, Real-Time Heuristic Search: New Results, *Proceedings AAAI-88, St. Paul, Minnesota*, AAAI, 1988, 139–144.

[37] Laird, J.E., Newell, A., and Rosenbloom, P.S., SOAR: An Architecture for General Intelligence, *Artificial Intelligence*, **33** (1987) 1–64.

[38] Mendenhall, W., *Introduction to Linear Models and the Design and Analysis of Experiments*, (Wadsworth, Belmont, California, 1968).

[39] Minton, S., Carbonell, J. G., Knoblock, C. A., Kuokka, D. R., Etzioni, O., and Gil, Y., Explanation-Based Learning: A Problem Solving Perspective, *Artificial Intelligence*, **40** (1989) 63–118.

[40] Pearl, Judea, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, (Morgan-Kaufman, Los Altos, California, 1988).

[41] Raiffa, Howard and Schlaifer, R., *Applied Statistical Decision Theory*, (Harvard University Press, 1961).

[42] Rosenschein, Stan, Synthesizing Information-Tracking Automata from Environment Descriptions, In Brachman et al. [6], 386–393.

[43] Russell, Stuart J. and Wefald, Eric H., On Optimal Game-Tree Search using Rational Meta-Reasoning, *Proceedings IJCAI 11, Detroit, Michigan*, IJCAI, 1989, 334–340.

[44] Russell, Stuart J. and Wefald, Eric H., Principles of Metareasoning, In Brachman et al. [6].

[45] Shachter, Ross D., Evaluating Influence Diagrams, *Operations Research*, **34**(6) (1986) 871–882.

[46] Simon, Herbert A. and Kadane, Joseph B., Optimal Problem-Solving Search: All-or-None Solutions, *Artificial Intelligence*, **6** (1975) 235–247.

[47] Smith, David E., *A Decision-Theoretic Approach to the Control of Planning Search*, Report No. LOGIC-87-11, Stanford Logic Group, 1988.

[48] Weyhrauch, R. W., Prolegomena to a Theory of Mechanized Formal Reasoning, *Artificial Intelligence*, **13** (1980) 133–170.