Reasoning Under Varying and Uncertain Resource Constraints[*]

Eric J. Horvitz
Medical Computer Science Group
Knowledge Systems Laboratory
Stanford University
Stanford, California 94305

Abstract

We describe the use of decision-theory to optimize the value of computation under uncertain and varying resource limitations. The research is motivated by the pursuit of formal models of rational decision making for computational agents, centering on the explicit consideration of preferences and resource availability. We focus here on the importance of identifying the multiattribute structure of partial results generated by approximation methods for making control decisions. Work on simple algorithms and on the control of decision-theoretic inference itself is described.

1 Computation Under Uncertainty

We are investigating the decision-theoretic control of problem solving under varying constraints in resources required for reasoning, such as time and memory. This work is motivated by the pursuit of formal models of rational decision making under resource constraints and our goal of extending foundational work on normative rationality to computational agents. We describe here a portion of this research that centers on reformulating traditional computational problems into strategies for generating and reasoning about a spectrum of *partial results* characterized by multiple dimensions of value. After describing work on the solution of classical problems under uncertain and varying resource constraints, we shall return briefly to the larger, motivating problem of computational rationality, focusing on the pursuit of optimal strategies for computing beliefs and actions under resource constraints.

A rational agent applies an inference strategy with the intention of performing an analysis that will be of some net benefit. There is usually uncertainty about the best way to solve a problem because of incompleteness in knowledge about (1) the value of alternative computed results in a particular situation, (2) the difficulty of generating results from a problem instance, and (3) the costs and availability of resources (such as time) required for reasoning. We have been investigating the use of decision theory for valuating alternative problem-solving strategies under uncertainty. Thus, we define components of computational value in terms of expected utility [7]. The use of decision theory to guide the allocation of computational effort was proposed by Good several decades ago [3].

1.1 Computational Utility

We use the term *computational utility*, u_c, to refer to the *net* value associated with the commitment to a computational strategy. We decompose u_c into two components: the *object-level* utility, u_o, and the *inference-related* utility, u_i. The *object-level* utility of a strategy is the benefit attributed to acquiring a result, without regard to the costs associated with its computation. For example, the object-level utility of a medical expert-system inference strategy is the value associated with the information it generates about the entities in a medical problem, such as alternative treatments and likelihoods of possible outcomes. The *inference-related* component, u_i, is the cost of the reasoning. This includes the disutility of delaying an action while waiting for a reasoner to infer a recommendation.

The decomposition of computational utility focuses attention explicitly on the costs and benefits associated with problem-solving activity. In the general case, we must consider the dependencies between the object- and inference-related value. We assume the existence of a function f that relates u_c to u_o, u_i, and additional information about the problem-specific dependencies that may exist between the two components of value—that is,

$$u_c(\alpha, \beta, \gamma) = f[u_o(\alpha, \gamma), u_i(\beta, \gamma)]$$

where α and β represent parameters that influence respectively the object- and inference-related utilities and γ represents the parameters that influence both the object- and the inference-related utilities.

1.2 Multiple Attributes of Utility

In real-world applications, the object-level and inference-related utilities frequently are functions of multiple attributes. Dimensions of value can be acquired through consultation with computer users. Computational utility may

[*]This work was supported by a NASA Fellowship under Grant NCC-220-51, by the National Science Foundation under Grant IRI-8703710, and by the National Library of Medicine under Grant RO1LM0429. Computing facilities were provided by the SUMEX-AIM Resource under NIH Grant RR-00785. Published in *Proceedings of the Seventh National Conference on Artificial Intelligence*, Minneapolis, MN. August 1988. Morgan Kaufmann, San Mateo, CA. pp. 111-116.

be assessed as numerical quantities for particular outcomes, or may be described by a function that represents the relationships among costs and benefits associated with alternative outcomes. Such functions assign a single utility measure to computation based on the status of an n-tuple of attributes. Let us assume that we can decompose u_c into u_o and u_i. A set of object-level attributes, $v_{o_1}, \ldots, v_{o_n}$, captures dimensions of value in a result, such as accuracy and precision, and defines an object-level attribute space, $\mathcal{A}_o$. A sequence of computational actions, c, applied to an initial problem instance, I, yields a result, $\phi(I)$, that may be described as a vector $\vec{v}_o$ in $\mathcal{A}_o$. Components of the inference-related cost—such as the computation time, memory, and, in some applications, the time required to explain machine reasoning to a human—define a resource attribute space, $\mathcal{A}_r$. In this paper, we simplify $\mathcal{A}_r$ to r, the scalar quantity of time. If we assume that u_o and u_i are combined with addition and $u_i(r)$ is the cost of delay, we can say that

$$u_c(\vec{v}_o, r) = u_o(\vec{v}_o) - u_i(r)$$

2 Toward a Continuum of Value

Much of work on the analysis of algorithms has been directed at proving results about the time required for computing a solution defined by simple goals and termination conditions [1]. Although this perspective imposes useful simplification, it has biased synthesis and analysis toward solution policies that are indifferent to variation in the utility of a result or to the costs and availability of resources. We wish to increase the value of computation under limited and varying resources by identifying and characterizing classes of approximate or *partial* results that can be produced for a fraction of the resources required by the best available methods for generating final results.

Let c refer to a sequence of primitive computational actions. We define a subclass of sequences of computational actions, c_ϕ, that transform a specific problem instance I (e.g., a randomly mixed file of records) into a final result, $\phi(I)$ (e.g. a total ordering over records), without assistance from an omniscient oracle, $c_\phi[I] \rightarrow \phi(I)$. We define $r(c_\phi, I)$ as the resource required by c_ϕ to generate $\phi(I)$ from I. A majority of traditional algorithms generate specific c_ϕ given I, halting upon reaching a queried $\phi(I)$.

2.1 Partial Results

Wide variations in the value of a result to an agent, u_o, in the availability of r, and in the cost $u_i(r)$ suggest that the focus on time complexity for termination on final results is limited; analyses centering on how *good* a solution can be found in the time available can be crucial. The traditional approach is based, in part, on a tendency to assign only one of two measures of utility to computational behavior: either a final solution can be computed, which has maximum object-level utility, $u_o(\phi(I))$, or a solution is not found in the time available and the effort is considered a worthless expenditure. However, we can often construct sequences that produce approximate or *partial results* that have some fraction of $u_o(\phi)$.

We introduce flexibility into computation by defining another class of computational actions, c_π, that operate on instances, I, to produce partial results, $\pi(I)$, often requiring a fraction of the reasoning resources needed by c_ϕ to generate $\phi(I)$. That is, $c_\pi[I] \rightarrow \pi(I)$ and $r(c_\pi, I)$ is the resource required by c_π to generate $\pi(I)$. Partial results may be viewed as transformations of desired final results along one or more dimensions of utility where

$$0 \leq u_o(\pi(I)) \leq u_o(\phi(I))$$

and where u_o maps a real-valued object-level utility to attributes of $\pi(I)$ and $\phi(I)$. That is, in the context of a query for $\phi(I)$, $\pi(I)$ has object-level utility less than or equal to the utility of $\phi(I)$. However, reasoning costs can make c_π preferable to all available c_ϕ in particular contexts.

We associate with each partial result a vector in the space $\mathcal{A}_o$ for $\phi(I)$. For the purposes of summarization, it can be useful to define a context-independent distance metric $D : \mathcal{A}_o \times \mathcal{A}_o \rightarrow \mathcal{R}$ between points in this space. We relate the difference in utility of $\phi(I)$ and $\pi(I)$ to a function of the context and this distance. In general, however, the most meaningful distance metric is the difference in utility itself, $u_o(\phi(I)) - u_o(\pi(I))$. An example of a widely-used, context-independent distance among results is the numerical approximation, where D is a simple unidimensional measure of precision (e.g., the result of a Taylor series carried to a particular term). In this case, $\phi(I)$ and $\pi(I)$ are separated by a distance in the space of reals.

2.2 More Sophisticated Partial Results

We can move beyond the familiar numerical approximation to consider cases where D represents the divergence of $\pi(I)$ from $\phi(I)$ along higher-dimensional and more abstract properties of a computational result. Some classes of more sophisticated partial results are well-known. Others suggest new directions for research on reasoning under resource constraints. Dimensions in $\mathcal{A}_o$ often are based on the end use of the result and reflect human preferences.

Richer partial results include the output of Monte Carlo simulation methods. These methods partially characterize a probability of interest through probabilistically visiting portions of a problem; they yield a sequence of probability distributions over a set of states with additional computation. Another class of partial result is generated by randomized approximation algorithms. These results are statements of the form *the probability that the divergence of $\pi(I)$ is greater than x from $\phi(I)$ is less than y.* Randomized algorithms may produce valuable partial results in response to queries for $\phi(I)$ ranging from bin packing to probabilistic entailment. Within research on classical algorithmic problems, we can move from the traditional analysis of results at completion—such as sorting records in a file—to an examination of the manner in which alternative strategies refine valuable dimensions of a partial result as additional resource is expended. The manipulation of partial results and alternative approximation strategies is essential in reasoning about beliefs and actions under varying resource constraints. As examples, partial results may be generated by increasing the abstraction of propositions or by decreasing the completeness of dependencies in a decision model. It may even be useful to develop a metric that represents a distance in a conceptual space describing properties of inference. For example, a component of value might be the probability that a result will be consistent with an axiom or with a set of axioms.

2.3 Named Computational Strategies

To manage the complexity of computing, computer scientists have defined and characterized computation in terms of *strategies*. These computational policies include the familiar "named" sorting and searching algorithms. A strategy, $\mathcal{S}$, typically is defined as some repetitive pattern of computational activity in conjunction with a set of simple control rules that apply to a large class of inputs. A majority of strategies generate intermediate states that have no object-level value and that terminate when a specific queried $\phi(I)$ is produced. We use $\mathcal{S}_\phi$ to refer to such strategies. Partial-result strategies, $\mathcal{S}_\pi$, have an ability to generate transformations c_π. The iterative nature of many of these strategies allows us to represent the result produced by a strategy as a function of the problem instance and the amount of resource applied—that is, $\mathcal{S}_\pi(I, r) = \pi(I)$. We can endow $\mathcal{S}_\pi$ with termination criteria based on the costs and benefits of continuing to compute.

Several subclasses of $\mathcal{S}_\pi$ have the ability to refine attributes of object-level value as a continuous or bounded-discontinuous[1], monotonically increasing function of allocated resource. These *incremental-refinement* policies yield immediate object-level returns on small quantities of invested computation, reducing the risk of dramatic losses in situations of uncertain resource availability. The availability of a continuous range of partial results over some range of resource also grants control reasoners flexibility to optimize the tradeoff between $u_o(\pi(I))$ and $u_i(c_\pi, I)$ under varying object-level utilities and resource constraints. Particularly flexible *spanning* $\mathcal{S}_\pi$ converge on $\phi(I)$ and demonstrate continuous, monotonically increasing refinement as the applied resource ranges from zero to the quantity of resource required for convergence. It may be desirable for $\mathcal{S}_\pi$ to generate results that converge near or on $\phi(I)$ for quantities of resource less than or equal to the resources required by the most efficient known $\mathcal{S}_\phi$ to produce $\phi(I)$. Unfortunately, this may not be the case: an agent frequently must pay a resource penalty for having access to $\pi(I)$ at times before a preferred $\mathcal{S}_\phi$ could generate $\phi(I)$.

[1] *Bounded discontinuity* refers to a policy's ability to perform a specified ϵ refinement of one or more attributes in $\mathcal{A}_o$ for some δ expenditure of r, over a specified range of r. Other desirable properties for bounded-resource strategies are discussed in [5].

3 Uncertain Resources and Challenges

Issues surrounding computation under varying and uncertain resource limitations are being explored within the Protos project.[2] We seek to develop, at design time, inexpensive methods for selecting among strategies during real-time reasoning. We have been assessing prototypical utility and resource contexts for designing control decision rules. We are particularly interested in control rules that use a fraction of the available resource to examine the context and instance, and construct or select a valuable strategy.

3.1 Prototypical Classes of Resource Constraints

Several classes of functions describing u_i have been examined, including the *urgency*, *deadline*, and *urgent-deadline* situations. These cost functions are common in many real-world applications and are based in such universal interactions as lost opportunity and competition for limited resources. The functions vary in form depending on the nature and criticality of the situation.

Urgency refers to the general class of inference-related utility functions that assign cost as some monotonically increasing function of delay. The *deadline* pattern refers to cases where $u_i(r)$ is 0 or insignificant until a certain level of resource $r = t_h$ is reached. At this point, computation must halt, and the maximum object-level utility attained before the halt must be reported immediately. Otherwise, the result is worthless. The *urgent-deadline* requires consideration of both the cost and availability of time.

3.2 Rational Decisions about Computation

A rational computation-policy decision optimizes the computational utility, u_c. Most frequently, this optimization must be done under uncertainty. Thus, we wish to make use of probabilistic knowledge. By explicitly introducing uncertainty, we move the notion of a control reasoner from a *knower* to a *believer*, committed to making its best guesses about the strategy to apply, given a problem, a problem-solving context, and background state of information. We use ξ in the conditioning statement of probabilities to denote the dependence of belief on background information. ξ may include a computer scientist's beliefs about the performance of a strategy based on logical knowledge, on empirical experience with the policy, and on intuition. Such beliefs can be updated with empirical data as a system's experience grows.

The performance of a policy can be represented as a probability distribution over partial results generated by the policy given an instance and a quantity of resource. In Protos experiments, we assumed a set of prototypical contexts, each associated with specific object-level and

[2] *Protos* is a partial acronym for *p*roject on computational *r*esources and *t*rade*o*ff*s*.

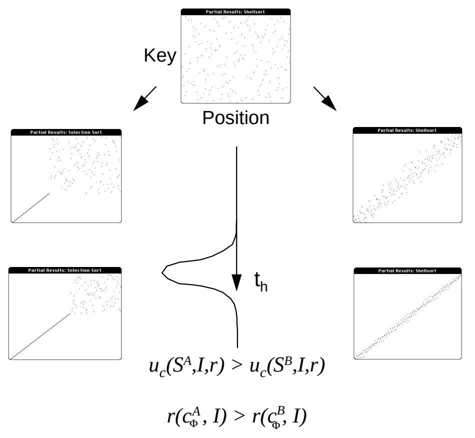
Partial Results: Shellsort
Key
Position
Partial Results: Selection Sort
Partial Results: Shellsort
Partial Results: Selection Sort
Partial Results: Shellsort
t_h
$u_c(S^A,I,r) > u_c(S^B,I,r)$
$r(c_\Phi^A, I) > r(c_\Phi^B, I)$

Figure 1: A graphical representation of incremental refinement by selection sort (left) and Shellsort (right).

inference-related utility functions. The computational utility of a partial-result policy in urgent situations is

$$u_c(\mathcal{S}_\pi, I, r) = \max_r \int_{\pi(I)} [u_o(\mathcal{S}_\pi, I, r) - u_i(r)]\; p[\mathcal{S}_\pi(I, r) = \pi(I)|\xi]$$

For valuating strategies limited to generating final results, this optimization considers the likelihood of generating the maximum object-level value, $u_o(\phi(I))$ over a range of resources. In urgent situations, a rational controller should choose the strategy $\mathbf{S}^*$ with the highest expected value,

$$\mathbf{S}^* = \arg\max_{\mathcal{S}} [u_c(\mathcal{S}, I, r)]$$

An agent immersed in a world of deadline situations must also grapple with uncertainty about the amount of time available for computation. Assume an agent has a probability distribution over the time available for computation in a situation. Given a set of strategies, what is the optimal strategy now? We first define the amount of resource that maximizes the expected utility of a policy,

$$r_{max}(\mathcal{S}, I) = \arg\max_r [u_c(\mathcal{S}, I, r)]$$

Then we consider cases where the deadline, t_h, occurs before r_{max} and cΔases where the deadline occurs after r_{max}. Under an urgent-deadline situation $\mathbf{S}^*$ is

$$\arg\max_{\mathcal{S}} \left[\int_{t_h < r_{max}} p(t_h|\xi) u_c(\mathcal{S}, I, t_h) + \max_r [u_c(\mathcal{S}, I, r)] \int_{t_h \geq r_{max}} p(t_h|\xi) \right]$$

In the pure deadline situation, we set $u_i = 0$, equivalent to substituting u_c with u_o in this equation. Similar integrations yield the utility for cases where knowledge about computation is encoded in terms of uncertainty in the resources required to generate specified results or where there is uncertainty or time-dependent variation in u_o or u_i.

4 Sorting Under Resource Constraints

Our research is directed primarily on the control of decision-theoretic inference. However, the problems have been generalized to other computational tasks. Here, we make use of the classic problem of sorting a file of records to present several issues in reasoning under varying resource constraints. Our analysis centers on identifying valuable dimensions of partial results, applying value functions that capture preferences about the results, and characterizing the ability of alternative strategies to refine results under certain or uncertain time constraints. We shall return briefly to problems of belief and action after exploring resource considerations with sorting algorithms.

4.1 Multiple Dimensions of Value

We constructed a prototype system, named Protos/Algo for exploring the value structure of alternative reasoning strategies. The system reports u_o, u_i, and u_c as a partial result is generated. To gain intuition and to help with the assessment of preferences, we have experimented with the graphical representation of partial results and partial-result trajectories. We used the system to probe the value structure of sorting algorithms.

We have defined alternative attribute spaces, $\mathcal{A}_o^{Sort}$, and explored the trajectories of the partial results produced by several named sorting policies. We experimented with several object-level and inference-related utility models that map points in the sorting space to computational utility.

Sample dimensions of value that may be useful in characterizing a partial sort include

- *Disorder*: the average distance between the current locations and expected final locations for records in a file or within specified portions of a file
- *High/low-end completion*: the contiguous length of positions, starting from the high (or low) end of the file, that contains records that are currently in the positions they will occupy after the file has been completely sorted
- *Bounded disorder*: an upper bound on the distance between the current position and final position for any record in a file

Other attributes can be formulated by working with the end user of a partial sort. For example, we can introduce an attribute representing the proportion of records that satisfy a particular level of *bounded disorder* or the probability that a partial sort will satisfy a specified value of *high-end completion* or *bounded disorder*. We could also seek to characterize the manner in which algorithms refine the values or probability distributions over attributes of interest. We can even extend an attribute such as *bounded disorder* to guide a search under resource constraints.

4.2 Alternative Trajectories Through a Multiattribute Space

The multiattribute nature of partial results adds additional richness to control decisions under resource constraints: The decisions can depend on the details of the problem-solving *trajectories* taken through the multiattribute partial result space. That is, there are different ways to refine a result with the application of resource. Alternative $\mathcal{S}(I)$ are associated with characteristic *patterns* of refinement. They may define distinct sets of points, curves or surfaces through $\mathcal{A}_o$ in response to the expenditure of r.

To help with visualizing refinement trajectories in sorting, Protos/Algo can display partial sorts, represented as a set of points within a Cartesian space, where the axes represent the index of an array and the value of the key to be sorted. As indicated in the sequences in Figure 1, a randomly mixed initial problem instance is transformed into alternative sequences of partial results, depending on the strategy applied. The left side of Figure 1 shows the partial result trajectories of a selection sort; on the right side, a Shellsort is pictured. The final result, $Sort_\phi(I)$, is represented by a diagonal line. Shellsort is striking in its ability to refine gracefully bounded disorder. Selection sort is efficient for refining low-end completion.

4.3 Sensitivity to Resources, Preferences, and Trajectories

We found that decisions about the best sorting policy to apply are sensitive to the availability and cost of resources, the nature of the object-level and risk preferences of an agent, and the structural details describing the refinement of results by strategies. Under uncertain and varying resource constraints, an algorithm with a slower completion time may be preferred to a more efficient algorithm. A utility analysis can demonstrate the comparative value of alternative sorting procedures for different combinations or weightings of the dimensions of partial sort described in Section 4.2 for prototypical resource contexts. In sample analyses, where I is the task of sorting a list of several hundred randomly arranged keys

$$r(c_\phi^{Select}, I) > r(c_\phi^{Shell}, I)$$

That is, the selection sort is less efficient in generating a total ordering. Yet, given a utility model that places high value on low-end completion, there exists a range of deadline times where the u_c of the selection sort dominates the faster Shellsort. Changes in the resources available or in the object- and inference-related utility functions can change the dominance. For example, diminishing the importance of low-end completion in the object-level utility u_o or increasing the importance of disorder, increases the expected utility of the Shellsort sort. The expected value of the Shellsort also is boosted as the distribution over the deadline time is skewed to greater values of r.

4.3.1 Utility of Continuity

Several sorting strategies continuously refine one or more object-level attributes of a partial sort. For example, Shellsort continuously refines disorder and selection sort refines completeness. In contrast, traditional versions of algorithms with $O(N \log N)$ complexity [8]—including mergesort, heapsort, and quicksort—do not make valuable intermediate results available, and thus may be dominated by the polynomial approaches under conditions of uncertain or poor resource availability, or high cost of reasoning.

In experiments comparing the graceful Shellsort to quicksort and heapsort on instances of several thousand randomly-arranged records, Shellsort could dominate the algorithms, even though $r(c_\phi^{Shell}, I) > r(c_\phi^{Quick}, I)$. We can see the usefulness of continuous refinement easily by inspecting the computational utility equations in Section 3.2. Although heapsort may have an $O(N \log N)$ runtime, if a deadline occurs at some time before completion $\phi(I)$, $u_o(\pi(I)) = 0$. In fact, u_i can make the wait costly. Thus, under resource constraints, a more valuable result can be generated by committing to a more conservative $O(N^{1.5})$, yet more graceful Shellsort.

5 Belief and Action Under Resource Constraints

Our research on sorting under resource constraints was undertaken to show the universality of resource-constraint issues and to gain insight about more sophisticated bounded-resource problem solving. We touch on these issues here to bring perspective to the sorting work. See [4] and [5] for additional discussion. We have focused on problems with the control of decision-theoretic inference for making recommendations about action within complex, high-stakes domains such as medicine and aerospace. Within such domains, the losses associated with suboptimal decisions tend to render simple satisficing approaches inadequate and provide incentive for optimizing computational utility.

5.1 The Complexity of Rationality

Since its inception forty years ago, decision theory has been accepted in several disciplines as a normative basis for decision making. Recent research has focused on the computational complexity of probabilistic reasoning, which lies at the heart of decision-theoretic inference. The work has centered on reasoning within directed graphs called belief networks [10]. Belief networks are special cases of more general graphical representations, called *influence diagrams*, that allow actions and utilities of alternative outcomes to be represented in addition to beliefs [6]. Several belief-network topologies have resisted tractable algorithmic solution. An example of a difficult class of problems is called the *multiply-connected* network. Inference with these graphs has been shown to be $\mathcal{NP}$-hard [2]. Problems in complex areas such as medicine often require representation with multiply-connected networks. Thus, rational beliefs and actions may demand intractable computation.

We are addressing the intractability of naive models of normative rationality by using decision theory at the metalevel to reason about the most valuable decision model and inference policy. There have been several discussions of the use of decision theory for reasoning about the value of analysis; for example, see [9]. In particular, we have directed our attention to the development of partial-result strategies for inferring the most valuable actions. A long-term dream, motivating research on Protos and related projects on automated decision-theoretic inference, is to construct an integrated system akin to a Macsyma for belief and action under resource constraints. Our current work on real-world problems centers on the use of decision analysis for designing control policies for decision-theoretic inference under constraints in the tissue-pathology lab (Protos/PF) and in the operating room (Protos/OR).

5.2 Partial-Result Strategies for Computing Optimal Action

Given a problem instance, composed of a belief network deemed to be a complete representation of a problem, and a specific query about a belief or action, we often can formulate an $\mathcal{A}_o$ that represents dimensions of value. We can apply intelligent control techniques in an attempt to maximize $u_c(Belief_\pi, I, r)$. We are exploring the generation of partial results through modulating the abstraction and completeness of an analysis. Techniques for modulating the completeness include the probing of an inference problem through directed or probabilistic search. These methods can produce probability distributions or bounds on probabilities of interest. We also can modulate the completeness of a belief network model by deleting the consideration of propositions or of dependencies among propositions. In addition, the model can be reformulated to report relevant qualitative answers. Finally, under severe time pressure, general default beliefs and policies may have more expected value than any new inference. Several partial-result strategies display interesting multiattribute trajectories with the commitment of additional resources. As in the sorting example, the structure of the trajectories of alternative strategies can influence the selection of an optimal reasoning strategy. See [5] for discussion of trajectories of belief refinement and for a view of default reasoning as a resource-constrained, partial-result strategy.

6 Discussion

Our experimentation and analysis have highlighted several issues about reasoning under varying and uncertain resource constraints. First, it appears that interesting dimensions of value in partial results have been overlooked; more attention has been directed on techniques for computing a targeted $\phi(I)$. There clearly is value in exploring the rich multidimensional structure of partial-result strategies. Rational decisions about computation, such as the selection of a new strategy or the decision to cease computing, can be sensitive to details of the timewise-refinement trajectories, to the object-level utility function, and to the uncertainties in the functions describing the cost and availability of reasoning resources. A wide range of computer-science research efforts may benefit by pursuing the development of reflective strategies that are sensitive to varying resource and utility conditions.

Strategies that continuously refine the value of partial results with time are desirable for reasoning in situations of uncertain resource availability because they can reduce losses and introduce additional flexibility into computational decision making. The new opportunities for decisions frequently translate into increased expected utility under resource constraints. The ability of incremental-refinement strategies to make intermediate problem-solving states available also can be useful for creating new policies from sequences of strategies (e.g., apply selection sort to bolster low-end completion efficiently and Shellsort to refine the bounds on disorder). A custom-tailored sequence of strategies for generating $\phi(I)$ or $\pi(I)$ will often have greater computational utility than do more general, predefined policies.

We can introduce even more flexibility into reasoning by moving the level of analysis from *strategies* to *actions* to consider control opportunities at the microstructure of computational activity. Although this task is more complex, the finer patterns of computation and control possible may enable a reasoner to generate more ideal refinement trajectories. Such research may also elucidate the control strategies implicit in familiar policies and stimulate the creation of more general, decision-theoretic strategies that could implement the familiar policies as special cases.

Identifying useful dimensions of utility in computation and examining the refinement of partial results as a function of invested resources can also direct attention to new classes of approximation. For example, there is opportunity for developing inexpensive strategies for transforming valueless, intermediate states of traditional $\mathcal{S}_\phi$ algorithms into valuable partial results or into states that can be handed-off to other methods by a control reasoner. For example, in the realm of sorting, such techniques could be useful for concatenating $O(N \log N)$ strategies, in reaction to a specific problem instance, intermediate states, or observed real-time problem-solving trends.

Although our current work centers on the construction of inexpensive policy-selection procedures, the best control strategies (i.e., the control strategies that maximize u_c) may be expensive. Clearly, the evaluation of the best policy according to the decision formulae in Section 3.2 involves costly searching; in practice, we limit the analysis to a tractable number of policy options out of an infinite number of possibilities and move expensive analysis to the design phase. Given a set of constraints on hardware, knowledge, and time, it may be beneficial for an agent to allocate a significant fraction of its scarce problem-solving resources to the metalevel analysis of a computational policy or control strategy. Expending effort to recognize a problem instance and context, to plan a solution, to monitor and redirect reasoning, and to coordinate these components of metareasoning may be important in customizing default control policies developed at design time or learned during resource-rich idle-time analyses. The possible optimality of expensive or varying allocation of resource for control brings to light significant questions about multilevel analysis that focus attention on reflective decision-theoretic architectures, strategies, and formalisms [4]. It also suggests that decision-theoretic inference may have to rely, at some point, on poorly characterized assumptions.

7 Summary

Preliminary analyses of the multiattribute utility structure of partial results suggest that endowing agents with knowledge about multiple dimensions of value in computed results can increase the expected utility of their problem-solving behavior. More generally, work on the decision-theoretic design of computational policies for several problem classes has highlighted the promise of developing techniques for maximizing the value of computation under constraints in knowledge, hardware, and time. Pursuing such bounded optimality in problem solving appears to be particularly important for developing agents that must act in dynamic, competitive, and high-stakes domains.

8 Acknowledgments

David Heckerman and Gregory Cooper provided useful comments on an earlier draft. I am grateful to Bruce Buchanan, John Breese, George Dantzig, Ronald Howard, Nils Nilsson, Stuart Russell, Edward Shortliffe, and Patrick Suppes for providing feedback on this research.

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *Data Structures and Algorithms.* Addison-Wesley, Menlo Park, CA, 1983.

[2] G.F. Cooper. *Probabilistic inference using belief networks is NP-hard.* Technical Report KSL-87-27, Knowledge Systems Laboratory, Stanford University, Stanford, California, May 1987.

[3] I.J. Good. A five-year plan for automatic chess. In *Machine Intelligence*, 2:89–118, London, Oliver and Boyd, 1968.

[4] E.J. Horvitz. *The decision-theoretic control of problem solving under uncertain resources and challenges.* Technical Report KSL-87-16, Knowledge Systems Laboratory, Stanford University, Stanford, California, February 1987; revised November 1987.

[5] E.J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of the Third AAAI Workshop on Uncertainty in Artificial Intelligence*, pages 429-444, Seattle, WA. Association for Uncertainty in Artificial Intelligence, Mountain View, CA, August 1987.

[6] R.A. Howard and J.E. Matheson. Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*, chapter 3, pages 721–762, Strategic Decisions Group, Menlo Park, California, 1981.

[7] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, Princeton, New Jersey, 1947.

[8] D.E. Knuth. *The Art of Computer Programming: Sorting and Searching.* Addison-Wesley, Reading, Massachusetts, 1973.

[9] J.E. Matheson. The economic value of analysis and computation. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4(3):325–332, 1968.

[10] J. Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29:241–288, 1986.