Models of Continual Computation

Eric Horvitz
Microsoft Research
Redmond, Washington 98025
horvitz@microsoft.com

Abstract

Automated problem solving is viewed typically as the expenditure of computation to solve one or more problems passed to a reasoning system. In response to each problem received, effort is applied to generate a solution and problem solving ends when the solution is rendered. We discuss the notion of *continual computation* that addresses a broader conception of *problem* by considering the ideal use of the idle time between problem instances. The time is used to develop solutions proactively to one or more expected challenges in the future. We consider analyses for traditional all-or-nothing algorithms as well as more flexible computational procedures. After exploring the allocation of idle time for several settings, we generalize the analysis to consider the case of shifting computation from a current problem to solve future challenges. Finally, we discuss a sample application of the use of continual computation in the setting of diagnostic reasoning.

Introduction

Computational problem solving is viewed traditionally as the automated solution of problem instances that begins when an instance is submitted for analysis and ends when a solution is rendered. Research on flexible, anytime procedures has extended the simple notion of termination of problem solving from that of generating a precise result to a process of incremental refinement. However, these algorithms also solve challenges as they are encountered in real time. In this paper, we describe work on *continual computation* that extends the definition of the problem instance to a continuing sequence of problems and that considers the ideal use of the periods of time that are traditionally viewed as idle time between problems. We develop policies for effectively harnessing idle time to develop solutions or partial solutions to challenges that may be encountered in the future.

We first review the use of knowledge about potential future instances under uncertainty and describe the limitations in the scope of our analyses. Following discussion of policies for allocating idle time, we generalize the analysis to consider the case of directing resources from solving a current challenge to a potential future challenge. Finally, we discuss a sample application of the use of continual computation in automated diagnostic reasoning.

Continual Computation

Computational systems are often used in environments where relatively large amounts of idle time are pierced by intermittent bursts of problem instances. As an example, consider the case of the interactive use of a diagnostic reasoning system that computes the probability of diseases in a patient or faults in a mechanical system and that makes recommendations for gathering additional information. The process of diagnosis and information gathering is often marked by periods of idle time between the inferences undertaken to revise a diagnosis in response to the entry of additional observations or test results.

We will explore the ideal allocation of idle-time resources for precomputing results in several contexts. Rather than pursuing the use of general optimization methods to allocate resources (Dantzig 1963), we seek to elucidate principles of continual computation and tractable policy-generation procedures for classes of prototypical problems. Our analyses hinge on a consideration of the likelihood of alternate forthcoming problems. Given information about the likelihood of future problem instances, ranging from detailed probability distributions to more qualitative orderings in terms of likelihood, we wish to develop policies for the ideal expenditure of idle time.

In related work, investigators have discussed the value of optimizing the performance of computational systems given resource constraints (Horvitz 1988; 1990; Breese & Horvitz 1990; Dean & Wellman 1991; Heckerman, Breese, & Horvitz 1989; Zilberstein & Russell 1995). Previous work has also addressed the set of

opportunities for developing methods for caching partial, approximate results and final, precise results so as to optimize the run-time behavior of system (Horvitz 1989). However, few specifics were given in that work on design principles and decision policies.

Minimizing Computational Delay

Assume we have access to exact or approximate information about the probabilities, $p(I|E)$, of seeing different problem instances I in the next period, given some evidence about the problem or environment situation E. Explicit probability distributions, as well as more qualitative orderings over the likelihood of future instances, can be learned from data or may be generated by a model of a system or environment. We will review an example of the latter in the discussion of a sample application in diagnostic reasoning. Gaining access or modeling the likelihood of future instances can range from trivial to difficult depending on the application. We will not dwell in this paper on alternative means for acquiring information about the likelihood of problem instances. Rather, we focus on the derivation of ideal policies that take as input likelihood information at any level of precision that is available.

Given access to probabilities of future challenges, how should an agent spend its idle time? We will limit our analysis to the subset of models of continual computation that address maximizing the timeliness or quality of the solution of the next problem challenge. We wish to identify efficiently the best allocation of resources to solve this greedy continual computation problem for canonical contexts, and, more generally, to identify some basic principles for harnessing idle time.

We first consider models for achieving the goal of minimizing the time required for a system to solve problems. We will assume that the time required to solve each problem is equal and that we have enough memory to store the partial results for the potential next subproblems, I. Our goal is minimize the expected delay when a challenge is posed to the system. Idle time ends at the moment that a new problem is received by the reasoning system.

The expected delay associated with solving the next problem instance is a function of the actions the system takes to precompute answers to problem challenges and the duration of the idle-time period. We use $t(I_i)$ for the time required to compute a solution to problem instance I_i, T for the total usable idle time, and t_i^f for the idle-time fraction allocated to computing the answer to problem instance I_i ahead of time. The maximal time that can be allocated to a future problem instance is the time needed to solve the problem, $t(I_i)$, and the total usable idle time T is less then or equal to the maximal idle time, $T^m = \sum_i t(I_i)$, sufficient to solve all potential future problems.

The expected delay before generating a solution to a future problem is,

$$\sum_j p(T = T_j) \sum_i p(I_i|E)[t(I_i) - T_j t_i^f] \quad (1)$$

where the idle-time, T, is indexed by its magnitude and $T_j t_i^f \leq t(I_i)$. The expected savings gained from performing idle-time computation is

$$\sum_j p(T = T_j) \sum_i p(I_i|E) T_j t_i^f \quad (2)$$

What can we say about optimizing an assignment of the set of fractions t_i^f of total usable idle-time resource, T to alternate problems? Let us consider the case for some constant amount of idle time, T. We can rewrite the expected savings in the next period as

$$T \sum_i p(I_i|E) t_i^f \quad (3)$$

which will be maximized for any value of T by maximizing the quantity $\sum_i p(I_i|E) t_i^f$.

Theorem 1 Idle-Time Partition. *Given an ordering over the probability of problem instances, $p(I_1|E) > p(I_2|E) > \ldots p(I_n|E)$, representing the likelihood that these problems will be passed to a program in the next period, the idle-time resource partition that minimizes the expected computation time in the next period is to apply all resources to the most likely problem until it is solved, then the next most likely, and so on, until the cessation of idle time or solution of all problems possible in the next period.*

Proof: Assume an ordering over the probability of the next problem instances

$$p(I_1|E) > p(I_2|E) > \ldots, > p(I_n|E) \quad (4)$$

Consider the case where resources are only applied to the most likely subproblem. The total savings are maximized when we maximize,

$$T \sum_i p(I_1|E) t_i^f = T p(I_1|E) \quad (5)$$

where $T t_i^f \leq t(I_i)$ and the total usable idle time T is $T \leq \sum_i t(I_i)$.

Now consider the case where some resource fraction x is diverted from solving problem I_1, and is applied to one or more of the other subproblems $I_2, \ldots, I_n$. We show that directing some portion x of the idle-time resource fraction from the most likely instance to the other problems must be less optimal than allocating

all of the resources to the most likely problem. That is, we show that

$$Tp(I_1|E) > Tp(I_1|E)(1-x) + T\sum_{i=2}^{n} p(I_i|E)t_i^f \quad (6)$$

Our goal reduces to showing that

$$xp(I_1|E) > \sum_{i=2}^{n} p(I_i|E)t_i^f \quad (7)$$

We know that $\sum_{i=2}^{n} t_i^f = x$, so we need to show that

$$p(I_1|E)\sum_{i=2}^{n} t_i^f > \sum_{i=2}^{n} p(I_i|E)t_i^f \quad (8)$$

As $p(I_2|E) > p(I_{i>2}|E)$, we know that

$$p(I_2|E)\sum_{i=2}^{n} t_i^f > \sum_{i=2}^{n} p(I_i|E)t_i^f \quad (9)$$

By definition, $p(I_1) > p(I_2)$. Thus, we know that

$$xp(I_1|E) > \sum_{i=2}^{n} p(I_i|E)t_i^f \quad (10)$$

for any x. Thus, the allocation of any resource to other subproblems is not as valuable as expending those resource on the subproblem that is most likely to be seen.

When problem instance I_1 is solved completely, it is removed from consideration and the same argument is made with the remaining $n-1$ problems and remaining idle time, $T - t(I_1)$. ■

Theorem 2 Idle-Time Partition Indifference. *If two or more of the most likely problem instances have equal likelihood, the expected computation time in the next period is minimized by partitioning resources to the equally likely instances in any configuration of fractions. If all problem instances are equally likely, we are indifferent about the allocation of idle-time resources.*

Proof: Consider the terms $i = 1, 2$ of $T\sum_i p(I_i|E)t_i^f$ for the two most likely problem instances. As $p(I_1|E) = p(I_2|E)$, we know that

$$Tp(I_1|E)t_1^f + Tp(I_2|E)t_2^f = Tp(I_1|E)(t_2^f + t_1^f) \quad (11)$$

Thus, the contribution to the expected value will be the same for any combination of allocations to problems of equal likelihood as long as the total partition to all of the problems with equal likelihood is unchanged.

Theorem 1 tells us that we must allocate idle-time resources to solving problems in order of their likelihood. Problems of equal likelihood are solved after problems with greater likelihood have been solved and removed from consideration. In this phase of precomputation, we can partition the resources among the instances of equal likelihood in any way without effect on the expected run-time savings. ■

Minimizing the Cost of Delay

In many settings, the cost of waiting for a computational result depends on context-dependent time criticality (Horvitz & Rutledge 1991; Horvitz & Barry 1995). Let us generalize the results on minimizing expected delay to minimizing the *expected cost* of delay. The generalization is based on an analysis similar to the one we used to identify policies for minimizing delay. We assume that we have for each future problem instance a time-dependent cost function, $Cost(I_i, t)$, that takes as arguments, an instance and the time delay required to compute each instance following a challenge. Beyond specifying time criticality as a function of the instance, we can employ a distinct context variable. Without precomputation, the expected cost of waiting for a response to the next challenge is,

$$\sum_i p(I_i|E)Cost(I_i, t(I_i)) \quad (12)$$

The contribution to the overall expected cost of the delay required to solve each future instance I_i is $p(I_i)C(I_i, t(I_i))$. Idle time $T \leq t(I_i)$ applied to precomputing instance I_i will reduce the expected cost of delay by $p(I_i)[Cost(I_i, t(I_i)) - Cost(I_i, t(I_i) - T)]$.

We wish to allocate the total usable idle time in a way that minimizes the expected cost. To identify an ideal continual-computation policy for the general case of nonlinear cost functions, we must employ search, or greedy analysis with small amounts of resource. However, general strategies can be constructed for specific classes of cost function. For example, consider the case where costs increase linearly with delay, $Cost(I_i, t) = C_i t$, where C_i defines a rate at which cost is incurred for each instance I_i. The component of the comprehensive expected cost contributed by the expected delay solving each instance I_i is $p(I_i)C_i t(I_i)$. Allocating idle time to precompute an instance diminishes the expected cost or increases the expected value at a constant rate of $p(I_i)C_i$. The expected value in the next period is maximized by allocating idle time to commence solving the instance associated with the greatest expected rate of cost diminishment, $p(I_i|E)C_i$, and to continue until it is solved, and then to solve the instance with the next highest expected rate, and so on, until all of the instances have been solved. In the next section, we will delve further into the use of

rates of refinement in identifying continual computation policies for flexible strategies.

Considering the Value of Partial Results

So far, we have considered the minimization of the expected time and cost to generate a final answer given a problem posed after some period of idle time. We now generalize the idle-time considerations to optimizing the expected value of a system's response for the case where we have access to one or more flexible algorithms with the ability to generate partial results $\pi(I)$ that have value to a user before a final, precise answer is reached.

Flexible Computation and Expected Value

Temporally flexible, anytime methods continue to enhance the value of partial results with computation. A system applies a flexible strategy S to refine an initial problem instance I or to further refine a partial result $\pi(I)$ stemming from prior computation (the partial result is a transformed problem instance I'). The expected value of computation (EVC) is the value of the refinement of a result with computation (Horvitz 1988). In the general case, it is important to consider the uncertainty of the results of computation, where computation generates a probability distribution over results,

$$S[\pi(I), t] \rightarrow p[\pi'(I)|\pi(I), S, t] \quad (13)$$

and, the EVC is,

$$\sum_j p[\pi'_j(I)|\pi(I), S_i, t] u_o(\pi'_j(I)) - u_o(\pi(I)) \quad (14)$$

where $u_o(\pi(I))$ is the object-level value of a previously computed partial result $\pi(I)$. For the case where cost is deterministic and separable from the value of computation, the net EVC (NEVC) is just

$$\text{NEVC}[S_i, \pi(I), t] = \text{EVC}[S_i, \pi(I), t] - C(t) \quad (15)$$

We consider the problem of allocating idle-time resources in terms of maximizing the expected value of the system when a problem instance is passed to the program in the next period problem-solving period. Rather than attempting to minimize the expected delay or cost for completely solving future problem instances, we attempt to maximize the expected value at the time a problem instance is received.

For simplification we assume that the selection of strategy S is predefined or optimized, and use S^* to refer to these strategies. The maximal expected value for the next period is maximized when the resource fraction assignment t_i^f is selected so as to optimize the expected value of precomputation (EVP),

$$\sum_j p(T = T_j) \sum_i p(I_i|E) EVC(S^*, \pi(I_i), T_j t_i^f) \quad (16)$$

In the general case, to optimize the policy on idle-time resource partition, we must consider the general problem of optimizing the allocation of resources under uncertain idle times. However, we can develop theorems analogous to Theorems 1 and 2 for special situations of EVC for a set of future problem instances.

The value of allocating resources to precomputing future problem instances in a system relying on flexible computation can be characterized in terms of the rate at which the best strategies can deliver value with computation. We use *EVC flux* to refer to the rate of change of value with computation time. The EVC flux, $\psi(S, I, t)$, for a problem instance I_i and strategy S is the instantaneous rate, $\frac{dEVC}{dt}$, at which the strategy delivers value at t seconds into solution of the problem.

In the general case, a strategy applied to a problem instance may deliver value as a nonlinear function of computational effort. We first consider the special case where computational strategies generate a constant EVC flux.

Theorem 3 Idle-Time Partition for Constant EVC Flux. *Given problem instances I_i that may be passed to a program in the next period, and an* EVC flux *$\psi(S, I_i, t)$ for the solution of each instance that is constant with time, the idle-time resource partition policy that maximizes the expected value at the start of the next period is to apply all resources to the problem with the maximal product of probability and EVC flux. That problem should be refined until a final result is reached, then the result with the next highest product should be analyzed, and so on, until the cessation of idle time or solution of all problems possible in the next period.*

Proof: We assume that allocation of time to each instance for preselected reasoning strategies, S^*, applied to problem instances provide constant EVC fluxes $\psi(S, I_i, t) = \psi_i$ for each instance based on the refinement of a sequence of partial results. We know that the EVP is,

$$\sum_j p(T = T_j) \sum_i p(I_i|E) EVC(S^*, I_i, T_j t_i^f) \quad (17)$$

and that,

$$EVC(S^*, I_i, T_j t_i^f) = \int_0^{T_j t_i^f} \psi_i dt = C_i T_j t_i^f \quad (18)$$

Thus, the EVP can be rewritten as,

$$\sum_j p(T = T_j) \sum_i p(I_i|E) C_i T_j t_i^f \quad (19)$$

For any amount of idle time T_j, less than the time required to solve all of the future instances, the fastest that the EVP can grow is by the instance that maximizes $p(I_i|E)C_i$. The ideal policy is to apply all resources to the instance with the highest value of $p(I_i|E)C_i$. Citing the same argument used in Theorem 1, any amount of time x re-allocated to another instance would diminish the total EVP because it would be multiplied with smaller valued products.

When problem instance I associated with the largest product, is solved completely, it is removed from consideration and the same argument is made with the remaining $n - 1$ problems. ■

Theorem 4 Idle-Time Partition Indifference for Constant EVC Flux. *If two or more of the most likely problem instances have an equal product of likelihood and EVC flux, the expected computation time in the next period is minimized by partitioning resources to the equally likely instances in any configuration of resource fractions. If all problem instances have equal product of likelihood and EVC flux, we are indifferent about the allocation of idle-time resources.*

The proof follows analogously to the proof of Theorem 2 with substitution of products of likelihood and EVC flux for the likelihoods.

Greedy Time-Slicing for Nonlinear EVC

For the general case of nonlinear EVC under uncertain idle time, we are forced to perform general optimization to seek optimal policies. However, we can derive approximate, greedy allocation strategies that take advantage of the results we described earlier. In the greedy, myopic approach, we consider the best allocation of small slices of the usable idle-time, Δt. At each stage, we consider the contribution to the total EVP for the allocation of Δt resources to solving each of several instances,

$$p(I_i|E)EVC(S^*, I_i, \Delta t) \quad (20)$$

We allocate all of the time to the instance with the greatest product of likelihood of seeing the problem instance, $p(I_i|E)$ and the EVC for the allocation of Δt, and continue to apply this greedy procedure as more time is available.

We justify the allocation of all of the resource in the Δt slice to the solution of a single instance by arguing that the EVC flux in small Δt regions is approximately constant, and employing Theorem 3 with the substitution of the average EVC flux during the time slice for the constant flux, C_i.

Considering the Cost of Shifting Attention

Costs may arise in computational systems based in the overhead of shifting attention from one problem to the next. Such costs can influence decisions about shifting to problems with greater EVC flux. Given the presence of costs of shifting attention, idle-time should be switched to refining an instance that yields a greater expected EVC flux only if the expected benefits of the switch are greater than the costs of the shift. To compute the expected net gain in shifting attention, we need to consider the probability distribution over remaining idle time, t^r, given idle time that has already been expended, t^e. Using $EVC_i(t)$ as shorthand for $EVC(S, I_i, t)$ and $Cost^s$ as the cost of shifting attention, the expected value of shifting to the new instance is,

$$\int_{t^r} p(t^r|t^e)[EVC_2(t^r)p(I_2) - EVC_1(t^r)p(I_1)]dt^r - Cost^s \quad (21)$$

As an example, consider the refinement of partial results for instances that are each represented by a concave-down piecewise linear EVC function. We select and apply idle time to solve the linear segment of the instance associated with the largest expected EVC flux. However, when we reach a segment with a diminished flux, we must reexamine the value of continuing to refine the current instance versus moving to another instance with a greater expected flux. Let us assume that the expected EVC flux for the next leg of refinement for the current instance, $p(I_1)$ is less than the expected EVC flux for another instance $p(I_2)$. That is, $p(I_1)C_1 < p(I_2)C_2$. The expected value of shifting to the new instance is,

$$\int_{t^r} p(t^r|t^e)t^r[C_2p(I_2) - C_1p(I_1)]dt^r - Cost^s \quad (22)$$

Equation 22 specifies that it will only be worth shifting if the mean remaining idle time, $\overline{t^r}$, is greater than the ratio of the cost of shifting attention and the difference of the expected rates,

$$\overline{t^r} > \frac{Cost^s}{C_2p(I_2) - C_1p(I_1)} \quad (23)$$

Beyond Idle Time: Trading the Present for the Future

So far, we have considered the allocation of idle time for solving future challenges and have assumed that

real-time resources would be dedicated solely to solving the current problem challenges. In some situations, allowing a portion of real time to be allocated to future problem solving with degradation of the quality or speed of solving the current challenges can enhance the value of a system's performance. It is ideal to allocate current resources to future problem solving when the expected value of the allocation to the future outweighs the expected gains of applying the resource in the present.

In making such tradeoffs, we must consider a user's preference regarding the net present value of results delivered by computational analysis of a future problem instance. Such a preference can often be captured by a time-discount factor that diminishes the value delivered at some later time to its net present value. In an analysis of the relative value of using computational resources to solve a problem in the present moment versus to apply the resources to solving future potential problems, we must consider the difference between the gains in the *discounted* value of future problem solving and the losses in current problem solving based in the transfer of resources.

The immediate loss of dedicating current resource to a future problem is the product of the quantity of time allocated and the average EVC flux over this period. The gains of allocating these resources to a future problem is a function of the amount of idle time that will be available after the current session is completed. If there will be sufficient idle time to precompute all possible future problem instances, nothing can be gained by the transfer of resources to precomputation. We must model the probability distribution over the available idle time, and use this information to compute the expected value of the transfer of resources to the future problems.

For the general case of nonlinear EVC flux on future problems, we must perform a general optimization. However, we can develop analyses for special cases. Let us consider the value of allocating a sequence of small slices of current time Δt to future problem solving. We assume a goal of maximizing the value of the result when the next challenge arrives. We assume that the net present value of enhancements delivered in the next period is the future value diminished by a constant time-discount factor, $d \leq 1$.

To compute the gain in making the transfer, we weigh the gain for each amount of idle time by the probabilities of idle time. We employ Theorem 3 to order the allocation of resources by the product of the constant EVC flux, ψ_i, generated by solving each problem and the likelihood of the problem, $p(I_i|E)$. We sort the future problem instances by the product of the likelihood of the instance and the EVC flux, and label these instances $I_1, \ldots, I_n$.

To compute the value of the precomputation, we consider the uncertainty about the flux that will be achieved with the resource, which will depend on the future problem that will be solved with the resource. The flux generated by the resource depends on the total usable idle time T and the total amount of time that has already been re-allocated to future computation, t_a. The current value of transferring current problem solving time Δt to precomputation is,

$$\sum_{T=0}^{t(I_1)-t_a} p(T)\psi_1 \Delta t p(I_1|E)d(T) + \sum_{T=t(I_1)-t_a}^{t(I_1)+t(I_2)-t_a} p(T)\psi_2 \Delta t p(I_2|E)d(T) + \ldots \sum_{T=\sum_{i=1}^{n-1} t(I_i)-t_a}^{\sum_{i=1}^{n} t(I_i)-t_a} p(T)\psi_n \Delta t p(I_n|E)d(T) \quad (24)$$

where the discount rate is a function of time in the future when the results of precomputation are realized.

In use, as the amount of total time allocated to solving a strategy grows so that $t_a \geq t(I_i)$, we remove the future problem (and associated term) from consideration, as that problem instance will have been completely solved by precomputation.

Theorem 5 Real-time dominance for constant EVC flux. *It is not worthwhile to allocate resources from current problem solving to future problem solving if the current constant EVC flux is greater than the greatest product of likelihood of future subproblem and its EVC flux.*

Proof: The proof follows from Equation 24 and Theorem 3. The largest contribution from future computation will come when the current resource is applied to precomputing the instance associated with the leading term in Equation 24. This term contributes when the usable idle time is less than the time required to solve the first problem instance. Assuming a time-discounting factor, $d \leq 1$, the value of precomputation will be nonpositive when the current EVC flux, $\psi_o \geq \psi_1 p(I_1|E)$ for any value of idle time and discount factor. ■

Thus, if the current EVC flux is greater than the leading product of EVC flux and likelihood of future instance, we do not need to consider the discount rate and the probability of idle time to make a decision about precomputation with the resources.

We can extend these results to consider situations beyond constant ψ. For example, consider the case where the EVC is increasing but the EVC flux is continuously diminishing, $\frac{d\psi_o}{dt} < 0$ (*i.e.*, a monotonically increasing, concave-down performance profile). In this case, we begin to do precomputation when the EVP becomes positive by Equation 24 and continue to check the EVP as new ψ_i are encountered following the solution of alternate future problem instances.

Example: Continual Computation for Value of Information

Models of continual computation hold promise for enhancing the use of a spectrum computational procedures including optimization of resource allocation in operating systems and transmission of information over limited bandwidth networks. For example, the methods can be harnessed in the prefetching of web pages. The methods can also enhance a variety of automated planning and decision-making tasks.

An example of the use of continual computation is the computation of the net expected value of information (NEVI) in a normative diagnostic reasoning system—a system that operates based on the principles of probability and utility (Henrion, Breese, & Horvitz 1992). Normative diagnostic systems compute the probability of states of one or more variables of interest (*e.g.*, diseases, faults) from observations (*e.g.*, findings) (Ben-Bassat *et al.* 1980; Heckerman, Horvitz, & Nathwani 1992; Shwe *et al.* 1991). The systems provide users with a probability distribution, $p(H|E, e, \xi)$, over one or more variables of interest H, in response to evidence e, in the context of previously observed evidence E, and background state of information ξ. An important component of many diagnostic systems is the identification, at any point in a diagnostic setting, of the best next observations or tests to perform, based on the net expected value of information.

The NEVI is the difference between the value of observing new information under uncertainty, and the cost of making the observation. To compute the value of information, the systems consider, for each observation, the expected value of the best decision in the world for each value the observation can take on. Then, the expected utility for each value is summed, weighted by the probabilities of seeing the different values, should the observation be made,

$$\text{NEVI}(e_x, E) =$$

$$\sum_k p(e_{x,k}|E)[\max_A \sum_j u(A_i, H_j)p(H_j|E, e_{x,k})]$$

$$-\max_A \sum_j u(A_i, H_j)p(H_j|E) - C(e_x, E) \quad (25)$$

where e_x is an unobserved variable and $e_{x,k}$ is the observation of state k when e_x is evaluated. The computation of the net expected value of information (NEVI) can impose noticeable delays for computation in decision-support systems depending on the platform and the problem being solved. Probabilistic inference in general graphical models like Bayesian networks and influence diagrams is NP-hard (Cooper 1990; Dagum & Luby 1993). The computation of NEVI, even for the case of the greedy analysis, requires, for each piece of unobserved evidence, probabilistic inference about the outcome of seeing the spectrum of alternate values should that observation be carried out. Thus, multiple computations for each potential test or observation must be considered.

NEVI is a natural candidate for continual computation. Typically, the observation of evidence requires effort and time, which provides variable amounts of idle time to the system for precomputation. Also, the NEVI computation includes computing $p(e_{x,k}|E)$, the probability of future observations, should an observation be made. We can use this information to control continual computation of NEVI (Horvitz & Peot 1996).

The probability of the next observation depends on the way recommendations for observations are presented to the user in a diagnostic system. Depending on the user interface of the diagnostic system, a user may be presented with only the best next observation to make or a list of observations to make ranked by NEVI. In the case of one recommendation, we use $p(e_x, k|E)$, the probability over future instances I. With the use of a list of recommended observations, we can employ a model of the probability of selection based on factors including the position of the recommended observation and/or relative NEVI values. We can compute the probability of the next problem instances as the product of the probability of the user selecting finding y from a list to evaluate, and the probability of the next observation given that selection, $p(y|\text{display}) \times p(e_x, k|E, y)$.

If we assume that the system will compute and cache the probabilities as well as the recommendations on the best future observations, that we have sufficient local memory to store the partial results of computation, and that problem instances are of equivalent difficulty, we can minimize the total time of computation in the next period by employing the policy of directing the available idle time to future NEVI problem instances in order of their computed probability.

Summary

We presented continual-computation policies for harnessing idle time to enhance the quality of solutions in

the next period, given information about the likelihood of instances. We considered procedures for minimizing delays and maximizing the quality of the response to forthcoming challenges. We reviewed procedures for ideally allocating idle time as well as current solution time to precompute future problems. The work highlights opportunities for leveraging "idle time" to enhance the expected value of a system's performance. We are exploring a variety of applications of continual computation in decision-making systems, networking, and operating systems. We are also pursuing extensions to the methodology, including issues that arise with the consideration of memory limitations and cost. Research on continual computation holds promise for enhancing the value of computational systems and for further bolstering our understanding of problem solving under bounded resources.

Acknowledgments

Natalia Moore, Jed Lengyel, Jack Breese, Chris Meek, and Mark Peot provided useful feedback on this work.

References

Ben-Bassat, M.; Carlson, V.; Puri, V.; Davenport, M.; Schriver, J.; Latif, M.; Smith, R.; Lipnick, E.; and Weil, M. 1980. Pattern-based interactive diagnosis of multiple disorders: The MEDAS system. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 2:148–160.

Breese, J., and Horvitz, E. 1990. Ideal reformulation of belief networks. In *Proceedings of Sixth Conference on Uncertainty in Artificial Intelligence, Cambridge, MA*, 64–72. Association for Uncertainty in Artificial Intelligence, Mountain View, CA.

Cooper, G. 1990. The computational complexity of bayesian inference using bayesian belief networks. *Journal of Artificial Intelligence* 42(2):393–405.

Dagum, P., and Luby, M. 1993. Approximating probabilistic inference in Bayesian networks is NP-hard. *Journal of Artificial Intelligence* 60(1):141–153.

Dantzig, G. 1963. *Linear Programming and Extensions.* Princeton, NJ: Princeton University Press.

Dean, T., and Wellman, M. 1991. *Planning and Control.* San Mateo, California: Morgan Kaufmann Publishers. chapter 8.3 Temporally Flexible Inference, 353–363.

Heckerman, D.; Breese, J.; and Horvitz, E. 1989. The compilation of decision models. In *Proceedings of Fifth Workshop on Uncertainty in Artificial Intelligence, Windsor, ON*, 162–173. Association for Uncertainty in Artificial Intelligence, Mountain View, CA.

Heckerman, D.; Horvitz, E.; and Nathwani, B. 1992. Toward normative expert systems: Part I. The Pathfinder project. *Methods of information in medicine* 31:90–105.

Henrion, M.; Breese, J.; and Horvitz, E. 1992. Decision analysis and expert systems. *AI Magazine* 12:64–91.

Horvitz, E., and Barry, M. 1995. Display of information for time-critical decision making. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 296–305. Montreal, Canada: Morgan Kaufmann, San Francisco, CA.

Horvitz, E., and Peot, M. 1996. Flexible strategies for computing information value in diagnostic reasoning. In *Fall Symposium on Flexible Computation, Cambridge MA, Report FS-96-06*, 89–95. AAAI: Menlo Park, CA.

Horvitz, E., and Rutledge, G. 1991. Time-dependent utility and action under uncertainty. In *Proceedings of Seventh Conference on Uncertainty in Artificial Intelligence, Los Angeles, CA*, 151–158. Morgan Kaufmann, San Mateo, CA.

Horvitz, E. 1988. Reasoning under varying and uncertain resource constraints. In *Proceedings AAAI-88, Minneapolis, MN*, 111–116. Morgan Kaufmann, San Mateo, CA.

Horvitz, E. 1989. Rational metareasoning and compilation for optimizing decisions under bounded resources. In *Proceedings of Computational Intelligence 89, Milan, Italy.* Association for Computing Machinery.

Horvitz, E. 1990. *Computation and Action Under Bounded Resources.* Ph.D. Dissertation, Stanford University.

Shwe, M.; Middleton, B.; Heckerman, D.; Henrion, M.; Horvitz, E.; Lehmann, H.; and Cooper, G. 1991. Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base–II: Evaluation of diagnostic performance. *Methods of information in medicine* 30:256–267.

Zilberstein, S., and Russell, S. 1995. Approximate reasoning using anytime algorithms. In Natarajan, S., ed., *Imprecise and Approximate Computation.* Kluwer Academic Publishers.