Thinking Ahead: Continual Computation Policies for Allocating Idle and Real-Time Resources to Solve Future Challenges

Eric Horvitz
Microsoft Research
Redmond, Washington 98052
horvitsOmicrosoft.com

Abstract

Research on *continual computation* centers on developing precomputation policies that can effectively harness available resources to solve future challenges. We focus on integrating a consideration of offline and real-time resources in continual computation. We review precomputation policies for flexible procedures and present strategies that account for the expected future real-time refinement of a result following precomputation. Finally, we address policies that consider the tradeoff between the efficiency of solving current and potential future challenges.

1 Introduction

Research on problem solving under bounded resources has focused primarily on real-time reasoning **[7, 8, 1, 2, 10, 3, 12, 11, 9, 14].** In contrast, work on continual computation centers on developing methods for harnessing the resources available in periods traditionally viewed as "idle time" between challenges **[4, 13].** Such policies have application to a variety of tasks ranging from generating contingency plans to prefetching web pages and computing ideal observar tions to make next in situation assessment or diagnosis. In this paper, we explore continual-computation policies that take into consideration both idle and real-time resources. The methods center on a consideration of the *expected value flux* generated by problem-solving strategies.

We first present several utility models that describe how strategies generate results of partial value with computation. For background and clarity, we shall review results described previously on ideal continual-computation policies for these utility models. Then we present a new class of precomputation policies by integrating a consideration of future real-time refinement that may follow the identification of a challenge. Finally, we examine the case of generalizing the methods to consider issues with trading the efficacy of real-time problem solving for enhanced future responses.

2 Utility of Precomputation

Assume we have access to probabilities, *p(I\E),* of seeing different problem instances / in the next period, given some evidence *E* that has been observed. Inferring the likelihood of future instances can range from trivial to difficult depending on the application. We will not focus in this paper on the means for building probabilistic models or collecting statistical data for inferring or accessing the likelihood of problem instances (see [5] for examples of learning probabilistic models for continual computation and [4] for a discussion of methods for computing likelihoods of future problem instances). Instead, we focus on the derivation of ideal policies that take as input likelihood information of any degree of precision about the occurrence of future problems. We seek to optimise 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 that have value to a user before a final, precise answer is reached.

A flexible or anytime reasoning strategy *S* has the ability to refine an initial problem instance / or further refine a partial result $\pi(I)$ that has been previously generated [6, 2]. The expected value of computation (EVC) is the change in utility with computar tion [8,11]. EVC is computed as follows:

$$\mathbf{EVC}[S_i, \pi(I), t] = \sum_j p[\pi'_j(I)|\pi(I), S_i, t] u_o(\pi'_j(I)) - u_o(\pi(I)) \quad (1)$$

where $\pi'(I)$ represents a refinement of $\pi(I)$, and $u_o(\pi(I))$ represents the *object-level* value of the initial or previously refined partial result $\pi(I)$, without consideration of the cost of computation. For the case where cost is deterministic and separable from the value of computation, the *net EVC* (NEVC) is just

$$\mathbf{NEVC}[S_i, \pi(I), t] = \mathbf{EVC}[S_i, \pi(I), t] - C(t) \quad (2)$$

where *C{t)* represents the cost of delay associated

from real-time computation, and use the phrase *expected value of precomputation* (EVP) to refer to the *exvccted* change in NEVC for actions taken to address challenges in a *future period.* The EVP associated with allocating offline resources, T, to refine the result associated with a potential future problem I is

$$\mathbf{EVP}(S_i, I, T) = p(I|E)\mathbf{NEVC}(S_i, I, T) \quad (3)$$

We can characterize flexible computation strategies in terms of the rote at which they deliver future value with precomputation. We refer to the instantaneous rate at which EVP changes at point of allocating *T* seconds of precomputation time to solving problem / with strategy 5, as the *EVP flux,* $\psi(S, I, T)$,

$$\psi(S, I, T) = \frac{dEVP(S, I, T)}{dT} \quad (4)$$

The EVP flux is the slope of the curve describing the utility associated with a computed partial result over time. For simplicity, we shall leave out the arguments of EVP and use $\psi(I, T)$ as shorthand to refer to the expected EVP flux associated with the strategy-instance pair after *T* seconds of precomputation. Let us assume for convenience that the selection of a strategy or sequence of strategies *S* is predefined or optimized for each problem instance, and use S* to refer to these strategies. We shall not dwell in this paper on the problem of choosing ideal strategies; such work has been a focus of research on real-time reasoning under varying and uncertain resources (e.g., see [2]). The choice of strategy does not influence the fundamental results on policies for continual computation.

Assume that we have access to probabilities and performance profiles that provide a measure of the EVP f $\psi(I, t)$, for each future problem instance *I* under consideration. Further, assume that a system apportions fractions of the total available idle time f_i to refining results for several potential future challenges. The overall EVP associated with multiple allocations can be computed by integrating the expected flux for resources allocated to each instance and summing together the EVP derived from each problem,

$$EVP = \sum_i \int_0^{Tf_i} \psi(I_i, T)dT \quad (5)$$

Given uncertainty in the amount of idle time, we have,

$$EVP = \int_0^T p(T) \sum_i \int_0^{Tf_i} \psi(I_i, T)dT \quad (6)$$

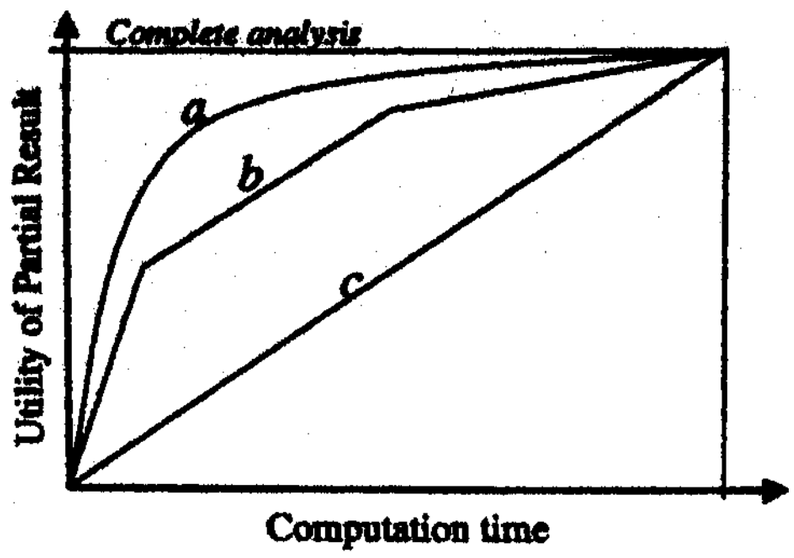

Figure 1: Prototypical classes of utility for partial results include (a) smoothly increasing utility with decreasing returns, (b) piecewise linear with decreasing returns, and (c) linear utility.

The goal of an automated reasoner endowed with the ability to reason about EVP is to leverage resources that are currently available to strategy-problem instance pairs so as to maximize the overall EVP. In the general case, identifying policies for allocating offline resources that yield the maximal future value, EVP*, requires a consideration of the details of the probability distribution over idle time, *p(T),* and the application of search or closed-form optimization methods to choose the best set of f_i. We seek precomputation principles, and associated policies, that forego the need for such optimization.

3 Policies for Prototypical Classes of Refinement

We shall review ideal policies for three classes of flexible procedures, defined by the functional form of utility models that describe the incremental value provided as they refine partial results. These classes include algorithms that refine results with *constant flux, piecewise linear flux with decreasing returns,* and *smoothly increasing utility with decreasing returns.* The utility models representing the performance profiles of these algorithms are displayed as representative curves in Figure 1.

3.1 Constant Flux

We first review ideal precomputation policies for the case of procedures that provide constant flux described previously in [4].

Theorem 1 Idle-Time Partition for Linear EVP Flux. *Given future problem* instances I_i *that may challenge a system in the next period, and an* EVP flux $\psi(I_i, t)$ *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 nest period is to apply all resources to the problem with the maximal EVP flux. That problem should be refined until a final result is reached, followed by the result with the nest highest product should be analyzed, and to on, until the cessation of idle time or solution of all problems possible in the nest period.

Proof: By definition, the allocation of time to each instance for strategies S applied to problem instances provide constant EVP fl $\psi(I_i, T) =$ foC_i each instance based on the refinement of a sequence of partial results. The total EVP can be rewritten as,

$$EVP = \int_0^T p(T) \sum_i C_i T f_i dT \qquad (7)$$

Fbr any amount of idle time 2\ the ideal policy is to apply all resources to the instance with the highest value of C_i. 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 / 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. ■

3.2 Prototypical Nonlinear Models

Policies for procedures yielding flux described by piecewise linear and smoothly increasing utility models can be viewed as a generalization of the case of constant flux. Instead of associating problem instances with constant levels of expected flux, we consider the flux associated with subcomponents of problems, and consider ideal strategies for solving these components. Continual computation policies for such models with a focus on the application of prefetching documents are described in [5]. We review the earlier results generalized to the case of real-time reasoning, before moving on to consider new policies that include a consideration of real-time resources.

Theorem 2 Idle-Time Partition for Piecewise Linear Utility with Decreasing Rate. *Given problem instances I_i, that may be accessed in the nest period, and EVP flux described by $\psi(I_i, T)$ that is piecewise linear and where successive segments have positive slope but progressively smaller flux, the resource partition policy that maximizes the expected value at the start of the nest period is to continue to allocate resources to the linear segment drawn from the set of nest available linear segments of instances that has the maximal expected flux, and to continue to refine the problem associated with that segment, and then*

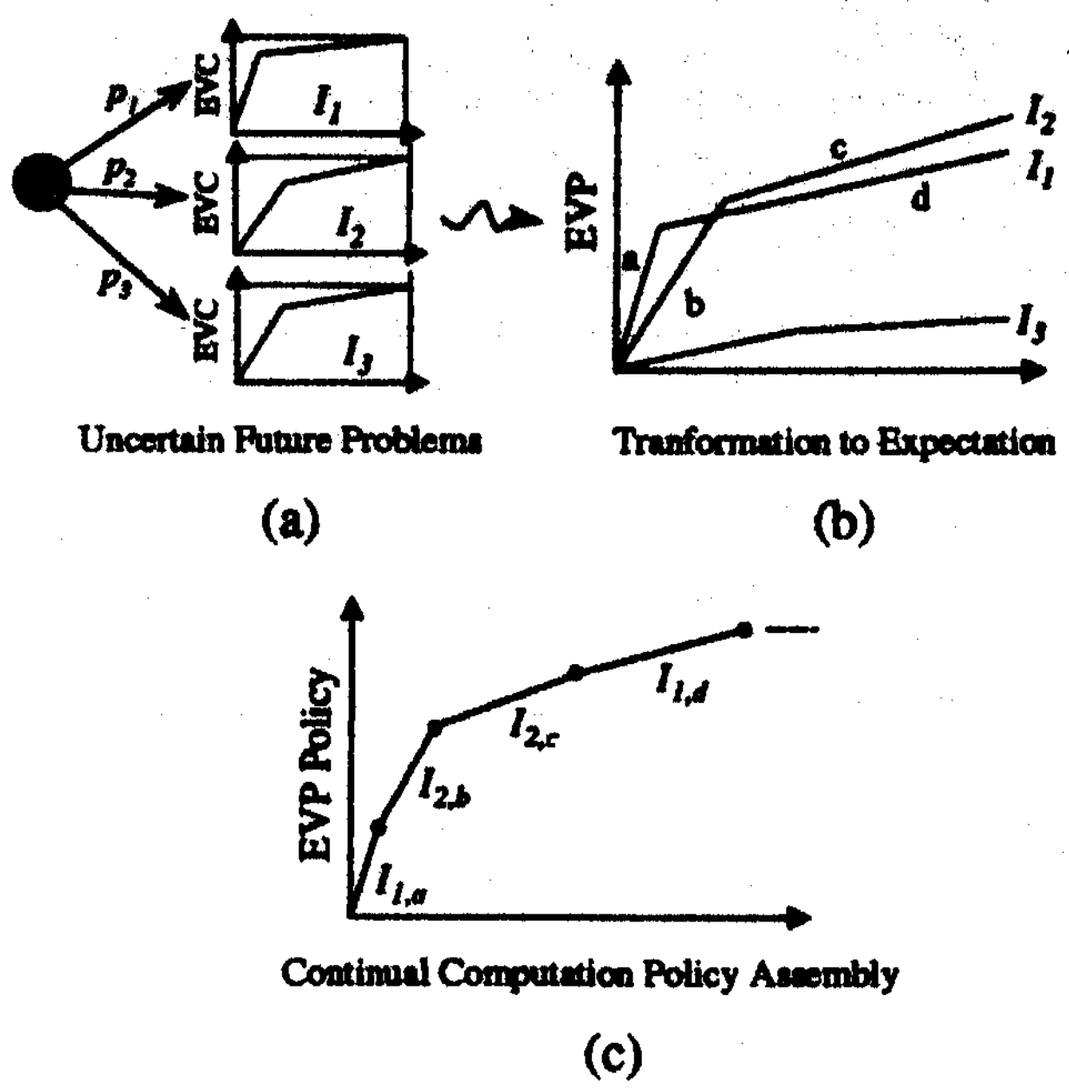

Figure 2: EVP analysis for case of piecewise linear utility with decreasing returns, (a) Consideration of future problems under uncertainty, (b) Mapping EVC to expectation, (c) Piecing together a continual computation policy considering expected flux of different segments.

to move to the segment with the next highest expected value flux until all segments of all problems are solved or the cessation of idle time.

Proof: We extend Theorem 1 on the ideal allocation of resources for constant flux to each piecewise linear segment associated with the current state of refinement of a problem. We now consider portions of problem solving associated with each segment of the piecewise linear utility function model instead of the entire problem. We know that the instance that offers the maximal instantaneous expected flux will add the most to the overall expected utility and that all other instances will deliver less value now and—by definition of decreasing returns—at all other times in the future. Including any other segment in the integration represented by Equation 6 would lead to a smaller total EVP for any amount of idle time. Thus, we need only to check the next available segment for each incompletely solved problem to identify the best sequence of segments. Choosing any other segments would lead to a diminishment of the overall expected value at the start of the next period as compared with this policy. ■

Thus, the policy for maximizing the contribution to the expected value in the next period will be to look

at all linear segments of all problems under consideration and to continue to solve the segment associated with the highest flux, then refine the problem associated with the segment with the next highest expected flux.

Figure 2 summarizes the approach for creating ideal EVP policies for the case of piecewise linear and smoothly increasing utility. We transform the fluxes associated with the computing each problem instance into *expected* fluxes for solving future problems given uncertainty in their occurrence. An ideal policy is identified by continuing to expend all resources to solve the instance that delivers the greatest instantaneous flux.

3.3 Smoothly Increasing Utility with Decreasing Returns

We can easily generalize Theorem 2 to the case of utility models represented as smoothly increasing functions by taking the size of segments in the piecewise linear models to zero in the limit.

Theorem S Partition of Resources for Smoothly Decreasing Flux. *Given problem instance-strategy pairs that may need to be addressed in the next period, and a value flux described by* $\psi(I_i, T)$ *for the solution of each instance associated with* $d\psi(I_i, T)/dt < 0$ *for the solution of each instance associated, the resource partition policy that maximizes the expected value at the start of the next period is to allocate resources to the problem with the maximal product of probability and instantaneous value flux, until all problems are solved or until the cessation of idle time.*

Proof: For situations where the expected flux is monotonically decreasing for all instances, the policy for maximizing the contribution to the expected value in the next period will be to continually pick the problem associated with the highest mean expected flux for any small quantity of expenditure. Because each instance has an expected flux that is monotonically decreasing with allocation of resources, the greatest currently available flux must be greater than the future expected flux associated with this or any other instance. Thus, any other order of analysis with inflnitesmal amounts of resource will result in a lower overall EVP, •

4 Influence of Future Real-Time Refinement

The EVP policies described above center on an optimization of the expected value of the system at the time a new challenge is received. We now extend the policies to consider real-time computation. We start with a consideration of the prototypical real-time problem of reasoning with a procedure that provides smoothly increasing utility with decreasing returns in the context of cost that increases constantly with time. A graphical representation of this problem is displayed in Figure 3. As displayed in the figure, the strategy provides increasing value, but with decreasing marginal returns for the same amount of computation at increasingly greater levels of refinement, until a final result is generated. We refer to this class of real-time reasoning problem as *decreasing return, constant cost* scenarios. We first identify the ideal quantity of real-time deliberation for these problems.

Theorem 4 Ideal Deliberation for Decreasing Returns, Constant Cost Scenarios. *For scenarios of decreasing returns and constant cost, action should be token immediately if the net EVC flux at the outset of a challenge is nonpositive; otherwise, real-time refinement should continue until the EVC flux is equal to the cost.*

A graphical depiction of the ideal halting time is represented in Figure 3. An instance is refined into partial results $\pi(I)$ with computation or precomputation time T, eventually reaching the final, completely solved result, $\phi(I)$. The EVC flux is equal to the rate at which cost is incurred at the ideal halting, t^*. At this time, the algorithm generates an ideal result, $\pi^*(I)$. Figure 3 shows the tangent to the utility curve at this point of refinement, representing the EVC flux at the ideal halting time. As displayed in the figure, the rate at which value is generated is balanced at this point of refinement by the rate at which cost is accrued. Continuing to perform real-time computer tion after this level of refinement is achieved would incur costs faster than gains in value.

Let us now consider EVP and continual-computation policies for problem solving captured by such decreasing returns, constant cost situations. We refer to the partial result generated at the ideal time indicated for haltint^*, for a particular cost function c, strategy S, and instance J, per Theorem 4 as the *ideal real-time result,* $\pi^*(I)$. The ideal real-time result is insensitive to the amount of resources allocated to precomputation of a problem instance because the rate at which cost is accrued is constant.

Applying idle-time resources to scenarios of decreasing returns with constant cost can be viewed as a means of achieving *cost-free refinement* Consider the value of precomputation for the case where a system knows that it will face a problem with probability 1.0. The NEVC at the time the ideal real-time result

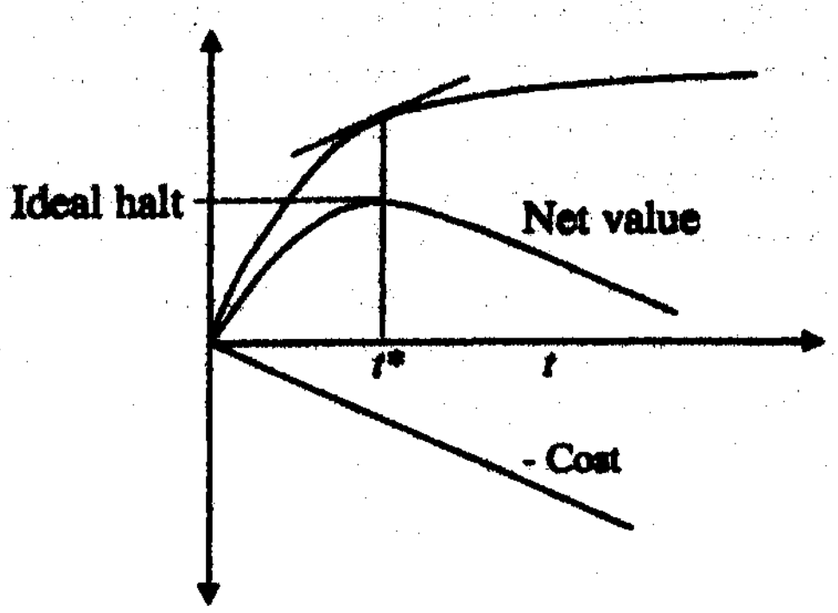

Figure 3: Graphical analysis of ideal halting time for real-time computation for the case of refining an instance with partial results that show decreasing returns in the context of constant cost. Cost is displayed on a negative scale for clarity.

is computed is simply the expected utility associated with an action in the world associated with $\pi^*(J)$, $eu(\pi^*(I))$, less the total cost accrued so far in computing the result. The total cost is simply the product of the rate at which cost is accrued and the amount of real-time resources expended on the problem prior to generating the ideal real-time result.

We can now analyze the value of precomputation in light of the prospect of future real-time reasoning to further refine a result. Precomputing a result of the quality represented by $\pi^*(I)$ *removes the cost function* for the time allocated to precomputing the result. Thus, the precomputation *adds value* to the ultimate utility that will be achieved when the result reaches the level of refinement represented by $\pi^*(I)$ with a flux equal to the rate at which real-time cost *would have been* accrued *without* precomputation. This flux is computed as the product of the probability of the instance and the cost.

Figure 4 displays graphically the impact of precomputation on the value of the result. The updated reasoning problem is represented by the solid lines portrayed in contrast to the broken lines, representing the situation before precomputation. As displayed in the figure, the cost function only begins to "tug" down on the net value of the result when real-time reasoning begins. Precomputation reduces the required real-time computation and raises the ultimate value of the ideal real-time result by the total amount of cost saved. Two small vertical arrows in Figure 4 represent the source and quantity of the jump in utility achieved with precomputation-

We now consider the case of continuing to refine the result with precomputation *beyond* the ideal real-time result. Additional precomputation applied to refining a result beyond $\pi^*(I)$ adds additional value to the

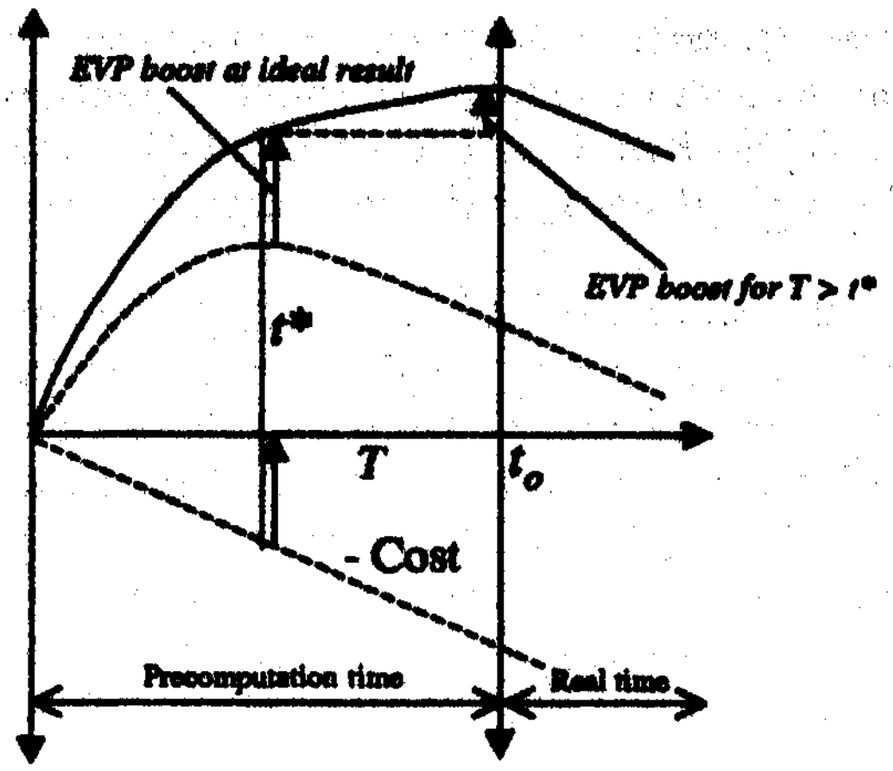

Figure 4: Graphical analysis of the influence of precomputation. Before the ideal real-time result is generated, a flux equal to the probability of the instance and the cost boosts the utility of the result. Additional precomputation boosts the result at the expected flux delivered directly by the strategy.

ultimate result at the cost-free rate associated with the *instantaneous flux* provided by the strategy in return for resources allocated beyond those required to achieve the ideal real-time result. Figure 4 demonstrates graphically how refining a result with precomputation *beyond* the ideal real-time result adds value in accordance with the flux delivered by the strategy in these regimes.

In summary, we must consider two situations for computing the EVP associated with precomputing instances for scenarios of decreasing returns, constant cost: If the level of refinement of a result reached with precomputation is less than the value represented by $\pi^*(I)$, the value flux is the product of the rate at which cost is accrued and the probability of being challenged with the instance. For refinement of a result by precomputation *beyond* the quality represented by $\pi^*(J)$, the value flux is the product of the probability of seeing the problem instance and the EVC flux associated with the strategy for increasing amounts of precomputation time following the achievement of $\pi^*(J)$.

Given our analyses of these two situations, we can develop policies for the general case of multiple problem instances and uncertainty. Given a set of potential future problem instances under uncertainty, we can adapt Theorem 4 to provide a policy for precomputation that takes into consideration future real-time reasoning. The policies consider all instances under uncertainty, noting for each instance whether the partial result achieved so for with precomputation fells short of the quality of the associated $\pi^*(I)$.

Theorem 5 Continual Computation Policy for Decreasing Returns, Constant Cost Scenarios. *For scenarios characterized by decreasing returns in a constant cost context, it is ideal to allocate offline computational resources to solving instances in order of the product of the probability of the instance and the cost of delay when refinement is below the utility associated with* $\pi^*(I)$ *and the product of the instantaneous rate of refinement and the probability of the instance when the value is greater than* $\pi^*(I)$.

Proof: For a constant probability of being challenged by an instance, the EVP flux for results of lesser quality than $\pi^*(J)$ is a constant determined as the product of the probability of the instance and the cost of delay. The EVP flux for results refined beyond the quality of $\pi^*(I)$ is the product of the probability and the instantaneous flux provided by precomputation. The EVP flux in regimes where quality is higher than $\pi^*(I)$ is smoothly decreasing and is smaller than the flux for refinement up to $\pi^*(I)$. Drawing upon Theorems 1 and 4, the maximal total EVP, will be obtained by continuing to select available instances with the highest expected EVP flux. Given that the EVP flux associated with strategies either provide constant or decreasing EVP flux with the allocation of precomputation time, any other order will lead to a suboptimal EVP. •

5 Redirecting Resources to Future Problems

So far, we have considered the allocation of available idle time solely for solving future problems. We have assumed that real-time computation is fully dedicated to solving a current challenge until reaching an ideal halting time. We can generalize the continual computation policies by considering the value of allowing real-time resources to be allocated to precomputation of *future* problems *before* a current problem is completed. Such a generalization considers trading a loss in the quality or timeliness of the response to current challenges for an enhanced expected response to future challenges. Such re-allocations can increase the overall expected value of a system's performance. It is worthwhile allocating resources from current to future problems when the EVP of potential future problems outweighs the EVC associated with solving the current problem.

The boost in EVP associated with the redirection of real-time problem solving to precomputing the solution of potential future problems depends on the amount of idle time that will be available before the next challenge arrives. If there will be sufficient idle time to precompute a majority of important future problem instances, little may be gained by an immediate transfer of real-time resources to precomputation* Thus, 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.

Let us use EVP*(r) to refer to the expected value of precomputation derived by following an ideal EVP policy (iVe., a policy dictated by the results described earlier), given the availability of T seconds of idle time between the completion of the current problem and the arrival of the next challenge. Suppose we are in the process of computing a response to a current challenge. The probability distribution over idle time, $p(T|E)$, can be determined from the probability distributions over the computation time required to complete the current analysis and over the time the next problem instance will be faced by the system. Assume that we have access to such probabilistic information, based on data collected on the performance of reasoning procedures and on the rates at which different classes of problem instances are faced by an agent in an environment or specific setting. Given $p(T|E)$, we compute the instantaneous change in EVP with computation of future challenges for different idle times T and sum over the uncertainty in idle time to generate an expected value flux at different values of idle time, $\frac{dEVP^*(T)}{dT}$,

$$\frac{dEVP}{dT} = \int_T p(T|E)\mathcal{D}\frac{dEVP^*(T)}{dT} \qquad (8)$$

where $\mathcal{D}$ represents a time-sensitive discount function that yields a discount rate, $d \leq 1$.

In real-time, we consider the instantaneous EVC flux associated with current problem solving with the instantaneous expected flux of solving future problems for small amounts of resource. If the current EVC flux is greater than the expected flux of solving future problems we continue to solve the current problem with the resource. However, if solving future problems has greater flux, we allocate the resource to computing future problems. We make the myopic assumption at each step that the probability distribution over idle time is determined by the time required to finish the current analysis. Thus, at each step of the analysis we consider an estimate of the revised time required for solving the current problem in 8. We can now build an overall EVP policy for allocating resources to each problem by considering the flux associated with each problem instance and continuing to choose the instance with the highest instantaneous flux.

6 Summary

We presented continual-computation policies for harnessing idle resources to enhance the future expected value of computation. We reviewed policies for several classes of refinement and developed an analysis and associated policy for folding in a consideration of additional refinement with future real-time resources. Finally, we described an approach to making decisions about redirecting resources being used solve a current challenge so as to precompute responses to potential future problems.

We are pursuing extensions to this work in several areas. Opportunities for ongoing research in continual computation include folding into the analyses a consideration of multiple periods in the future, reasoning about the explicit handling of sequences of challenges, and taking into consideration risk preference for handling uncertainty about the probability of future challenges.

Acknowledgments

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

References

[1] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings AAAI-88 Seventh National Conference on Artificial Intelligence*, pages 40-54. American Association for Artificial Intelligence, August 1088.

[2] T.L. Dean and M.P. Wellman. *Planning and Control*, chapter 8.3 Temporally Flexible Inference, pages 353-363. Morgan Kaufinann Publishers, San Mateo, California, 1991.

[3] D.E. Heckerman, J.S. Breese, and E.J. Horvitz. The compilation of decision models. In *Proceedinga of Fifth Workshop on Uncertainty in Artificial Intelligence*, pages 162-173. Association for Uncertainty in Artificial Intelligence, Mountain View, CA, August 1989.

[4] E. Horvitz. Models of continual computation. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 286-293. American Association for Artificial Intelligence, AAAI Press, Cambridge, MA, July 1997.

[5] E. Horvitz. Continual computation policies for utility-directed prefetching. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, pages 175-184. Association for Computing Machinery, ACM Press, New York, July 1998.

[6] E.J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of Third Workshop on Uncertainty in Artificial Intelligence*, pages 429-444, Seattle, Washington, July 1987. A A A I and Association for Uncertainty in Artificial Intelligence, Mountain View, CA.

[7] E.J. Horvitz. Reasoning under varying and uncertain resource constraints. In *Proceedings AAAI-88 Seventh National Conference on Artificial Intelligence*, pages 111-116. Morgan Kaufmann, San Mateo, CA, August 1988.

[8] E.J. Horvitz, G.F. Cooper, and D.E/Heckerman. Reflection and action under bounded resources: Theoretical principles and empirical study. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 1121-1127. Morgan Kaufinann, August 1989.

[9] S. Koenig and R.G. Simmons. Real-time search in nondeterministic domains. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1660-1667. Morgan Kaufinann, August 1995.

[10] R. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):18&-211,1990.

[11] S. Russell. Fine-grained decision-theoretic search control. In *Proceedings of Sixth Conference on Uncertainty in Artificial Intelligence.* Association for Uncertainty in Artificial Intelligence, Mountain View, CA, August 1990.

[12] S. Russell and E. Wefald. On optimal game-tree search using rational metareasoning. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence.* Morgan Kaufmann, August 1989.

[13] B. Selman, R.A. Brooks, T. Dean, E. Horvitz, T. M. Mitchell, and N.J. Nilsson. Challenge problems for artificial intelligence. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1340-1345. American Association for Artificial Intelligence, A A A I Press, Cambridge, M A, July 1996.

[14] S. Zilberstein and S.J. Russell. Approximate reasoning using anytime algorithms. In S. Natarajan, editor, *Imprecise and Approximate Computation.* Kluwer Academic Publishers, 1995.