ELSEVIER

Artificial Intelligence 126 (2001) 159–196

Artificial Intelligence

www.elsevier.com/locate/artint

Principles and applications of continual computation

Eric Horvitz

Microsoft Research, Redmond, WA 98052, USA

Received 12 January 1999

Abstract

Automated problem solving is viewed typically as the allocation of computational resources to solve one or more problems passed to a reasoning system. In response to each problem received, effort is applied in real time to generate a solution and problem solving ends when a solution is rendered. We examine *continual computation*, reasoning policies that capture a broader conception of *problem* by considering the proactive allocation of computational resources to potential future challenges. We explore policies for allocating idle time for several settings and present applications that highlight opportunities for harnessing continual computation in real-world tasks. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Bounded rationality; Decision-theoretic control; Metareasoning; Deliberation; Compilation; Speculative execution; Value of computation

1. Introduction

Research on inference and decision making under varying and uncertain resources has focused largely on computational methods for addressing real-time challenges. An implicit assumption in work on real-time problem solving is that reasoning begins when a problem is submitted for analysis and ends when a solution is rendered or action is taken in the world. Research on flexible, anytime procedures has extended the notion of termination of problem solving from that of generating a precise result to a process of incremental refinement. Nevertheless, flexible procedures have been studied primarily in the context of solving challenges as they are encountered in real time.

We shall move beyond real-time reasoning in pursuit of principles of utility-directed precomputation that we refer to as *continual computation* [40]. Continual computation

E-mail address: horvitz@microsoft.com (E. Horvitz).

0004-3702/01/$ – see front matter
PII: S0004-3702(00)00082-5

generalizes the definition of *problem* to encompass the uncertain stream of challenges faced over time. The methods bring attention to the potential value of developing machinery for reasoning in an incessant, context-dependent manner to forecast and prepare for potential *future* challenges [64].

There are compelling reasons for pursuing crisp policies for allocating idle resources to future real-time computation. Today, computational systems are typically employed in settings where relatively long spans of quiescence are pierced intermittently by bursts of tasks. Periods of inactivity are common in computer systems that are sporadically taxed to their limits when faced with real-time problem solving. At this moment, most of the computational resources in the world are going unused. Unless you are reading this document at some future date, when continual computation might be more ubiquitous, large numbers of operating systems are currently cycling in idle loops, waiting passively for the arrival of a real-time task.

The potential value of principles of continual computation for leveraging idle resources is especially highlighted when considering the challenge of creating autonomous inference and planning systems that are immersed in real-world environments, tasked with sensing and acting over prolonged periods of time. Such situated systems may frequently be in a resting state, but can expect to eventually face events and challenges that may lead to real-time computational bottlenecks.

Research on continual computation comes in the spirit of a body of work, centering on the value of optimizing the performance of reasoning and decision making under limited resources [7,11,22,26,27,29,43,46,48,51,59,63,68]. The work also shares motivations and goals with a variety of related efforts on compilation, precomputation, and prefetching in Computer Science.

Policies for guiding the precomputation and caching of complete or partial solutions of potential future problems are targeted at enhancing the expected value of future behavior. The policies can be harnessed to allocate periods of time traditionally viewed as idle time between problems, as well as to consider the value of redirecting resources that might typically be allocated to solving a definite, current problem to the precomputation of responses to potential future challenges under uncertainty.

Allocating idle resources to future problems is an intractable combinatorial optimization problem. In the general case, a combinatorial number of alternative allocations and sequences of precomputation effort must be explored and the best sequence selected, given a probability distribution over idle time. Rather than pursue the use of general optimization methods to allocate resources, we seek to elucidate principles of continual computation that can be harnessed to compose in a tractable manner ideal precomputation policies for prototypical classes of problems and scenarios.

We shall identify situations where local precomputation decisions lead to globally optimal precomputation. Our analyses hinge on a consideration of the probabilities of alternate forthcoming problems. Given information about the likelihood of future problem instances, ranging from detailed probability distributions to more qualitative orderings by likelihood, we seek policies that can guide the ideal expenditure of idle time.

We examine continual computation in several stages. First, we focus on continual computation for traditional, *all-or-nothing* problem solving. For such problems, results have no value unless a problem is solved completely. We consider the cases of minimizing

response time and of minimizing the losses associated with problem-specific costs of delay in addressing future tasks. Then, we explore continual computation for *flexible procedures* that have the ability to generate partial results of increasing quality until reaching a final, complete result. We explore several different situations of continual computation for these procedures, including the cases where we optimize the expected value at the start of a future period of problem solving and where we consider the additional real-time refinement and accrual of costs prior to a result being rendered or action being taken.

We introduce the notion of *expected value of precomputation (EVP) flux* as a unifying principle for continual computation and describe ideal policies for allocating idle and non-idle resources to precomputation, in terms of EVP flux. After laying out policies for several families of utility models for partial results, we discuss several challenges, including the cost of shifting attention from one instance to another, trading the efficacy of real-time problem solving for enhanced future responses, and precomputation and caching over multiple periods. Finally, we discuss illustrative applications of the use of continual computation, including its use in diagnostic reasoning and in the transmission of media under limited-bandwidth channels.

2. Minimizing computational delay

We shall assume a model of computation where a computing system is stochastically challenged with problem instances against a background of idle resources. We first define *problem instance*, *idle time*, and *precomputation time*.

Definition 1. A problem instance I is a task posed to a computing system in real time.

Definition 2. Idle time t^a is the period of time beginning at the completion or suspension of computation on the most recently posed problem instance and ending at the time a subsequent problem instance is posed to a computing system.

Definition 3. Precomputation time t^p is the amount of computation time applied to solving potential future problem instances.

Assume we have access to exact or approximate information about the probabilities $p(I \mid E)$ of seeing different problem instances I at the end of idle time, given some evidence E, representing observations at the beginning of an idle period about a context, situation, or environment, and background knowledge. Such probability distributions, as well as more qualitative orderings over the likelihood of future instances, can be learned from data or provided by a model of a system or environment.

Gaining access or modeling the likelihood of future instances can range from trivial to difficult depending on the application. We will review examples of drawing probabilistic information about future instances via computed and statistical models in the context of sample real-world applications in Section 11. Beyond discussion of models used in the sample applications, we will not dwell on the details for acquiring statistical information or building inferential models that provide likelihoods 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 on precomputation? We shall initially focus on the subset of models of continual computation that address maximizing the timeliness or quality of problem solving in the next period of problem solving.

We first construct a policy for minimizing the time required for computational systems to solve problems in a forthcoming problem-solving period. For this problem, we take as our goal the harnessing of idle time to minimize the *expected delay* following the posing of a new problem instance to a computing system. We shall assume that we have ample memory to store acutely the partial results for the set of potential subproblems $I_1, \ldots, I_n$ under consideration for precomputation.

The delay associated with solving a future problem instance is a function of the specific actions that the system takes to precompute responses to problem challenges and the overall duration of the idle-time period. We compute the expected delay by summing together the delay associated with solving each problem, weighted by the likelihood of the different problems.

We use t_i^p to refer to the precomputation time allocated to problem instance I_i and $t(I_i)$ to refer to the time required to compute a solution to the problem instance. We shall consider such times to be deterministic, but the analysis can be generalized to consider probability distributions by considering expectations for these times.

In the absence of any precomputation, the expected delay for solving one of the potential several future problem instances under uncertainty is

$$\text{Expected Delay} = \sum_i p(I_i \mid E) t(I_i). \tag{1}$$

If we apply computation to begin solving or to completely solve one or more potential problems, the expected delay in the next period is

$$\text{Expected Delay}' = \sum_i p(I_i \mid E)\big(t(I_i) - t_i^p\big), \tag{2}$$

where $t_i^p \leqslant t(I_i)$ and $\sum_i t_i^p \leqslant t^a$.

Theorem 1 (Policy for minimal expected time to completion). *Given an ordering over problem instances by probability $p(I_1 \mid E) > p(I_2 \mid E) > \cdots > p(I_n \mid E)$, that each of these problems will be passed to a problem solver in the next period, the idle-time resource policy 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 under consideration in the next period.*

We consider a set of instances, $I_1, \ldots, I_n$, ordered by the probability that each instance will be seen at the end of idle time. If we differentiate Eq. (2) with respect to precomputation time, we find that the rate of diminishment of the expected delay with the precomputation of each instance is just the probability of the instance, $p(I_i \mid E)$. Thus,

the greatest overall diminishment to the expected delay for all instances will be obtained by applying all resources to the most likely instance until that instance is solved. When problem instance I_1 is solved completely, it is removed from consideration and we make the same argument iteratively with the remaining $n-1$ problems and remaining idle time. Unless we are certain that we will have enough idle time to solve all of the problems under consideration, allocating resources to any problem except for the unsolved problem with the highest likelihood will lead to a summation representing a smaller diminishment in delay than allocating precomputation by problem likelihood. Thus, the ideal continual computation policy for minimizing the expected delay in the next period is to sequence the allocation of ideal resources by the likelihood of the problems until all of the instances are solved or a real-time challenge arrives.

This result tells us that if we are uncertain about having sufficient resources for solving all potential problem instances in the next period, we should allocate idle resources to problems in the order of their probability, regardless of the size of the problems.

We note that a system does not require precise probabilistic information to guide continual computation for minimizing delays at run time. A qualitative ordering over the likelihood of future problem instances provides us with all the information we need to construct an ideal policy for precomputation. Also, a system does not require complete knowledge about all potential future problem instances. We can represent an implicit set of unmodeled future instances by including an additional *unknown* state, I_{n+1}, in the problem instance variable, and assign an estimated probability mass and an expected required computation time for real-time solution of instances drawn from the set of unknown instances. Such implicitly represented sets of problem instances cannot be refined directly by precomputation as the instances remain unspecified. However, such probabilistic modeling of the unknown can be valuable in the coherent modeling and assignment of probabilities to future instances and for computing the overall expected value of applying continual computation policies.

Beyond specifying an ideal policy for minimizing real-time delays, we can make statements about idle-time allocation indifference. Because the rates at which delay is minimized with precomputation is equivalent for instances with the same likelihood, we can partition the resources among the instances of equal likelihood in any way without influencing the expected run-time savings. So, if two or more of the most likely unsolved instances have equal likelihood, we are indifferent to the partition of idle resources to these instances. If all problem instances are equally likely, we are indifferent about the allocation of idle-time resources.

3. Policies for minimizing the expected cost of delay

In time-critical situations, delays in generating a computational result are costly. For example, in the cases of high-stakes, time-pressured decision making in emergency medicine, physicians have to grapple with context-dependent costs of delayed evidence gathering and treatment [45]. We can generalize the results on minimizing expected delay to minimizing the *expected cost* of delay.

Let us assume that we have, for each future problem instance, a time-dependent cost function, $\text{Cost}(I_i, t)$, 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.

The contribution to the overall expected cost of the delay required to solve each future instance I_i is $p(I_i \mid E)\text{Cost}(I_i, t(I_i))$. Without precomputation, the expected cost of waiting for a response to the next challenge is,

$$\text{Delay Cost} = \sum_i p(I_i \mid E)\text{Cost}\big(I_i, t(I_i)\big). \tag{3}$$

Allocating idle resources $t^{p_i} \leqslant t(I_i)$ to the precomputation of instances I_i will reduce the overall expected cost of delay in the next period,

$$\text{Delay Cost}' = \sum_i p(I_i \mid E)\text{Cost}\big(I_i, t\big(I_i\big) - t_i^p\big), \tag{4}$$

where $\sum_i t_i^p \leqslant t^a$.

We wish to allocate the total usable idle time in a way that minimizes the expected cost. To identify an ideal policy for the general case of nonlinear cost functions, we are forced to 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, $\text{Cost}(I_i, t) = k_i t$, where k_i defines a rate at which cost is incurred for each instance I_i. In such situations, precomputation of each instance is associated with a constant rate of expected cost reduction, $p(I_i \mid E)k_i$.

Theorem 2 (Policy for minimal cost in constant loss settings). *Given an ordering of instances by the probability they will be seen in the next period and instance-sensitive losses that are linear with delays following the appearance of an instance, the policy that minimizes the expected cost in the next period is to apply available idle resources in order of the problem with the greatest rate of expected cost reduction until it is solved, then the instance with next highest rate, and so on, until the cessation of idle time or solution of all problems under consideration.*

We make an analogous argument to the discussion of Theorem 1. Here, the component of the comprehensive expected cost contributed by the expected delay for solving each instance I_i is $p(I_i \mid E)k_i t(I_i)$. Allocating idle time to precompute an instance diminishes the expected cost or, conversely, increases the expected value in the next period at a rate of $p(I_i \mid E)k_i$ for each instance. The overall expected value in the next period is maximized by allocating idle time to continue solving the instance associated with the greatest expected rate of cost reduction, 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. Beyond the case where a system has certain knowledge of sufficient idle resources to solve all of the instances, any other policy leads to the allocation of idle resources to solve an instance with a smaller rate when an instance with a larger rate is available, and, thus, is suboptimal for all situations.

Analogous to the case for minimizing expected future delays, we can make an assertion about allocation indifference for such situations. If two or more unsolved instances are associated with the same rate of expected cost reduction, we are indifferent to the partition of idle resources to these instances. If all problem instances have equal rates of expected reduction, we are indifferent about the allocation of idle-time resources to potential future instances.

4. Flexible computation, partial results, and precomputation

So far, we have described policies for minimizing the expected time and cost for algorithms with *all-or-nothing* outcomes. It is assumed that these algorithms must solve an instance completely before any value is derived from the computational investment. We now broaden considerations to consider flexible, anytime computational strategies—methods with the ability to refine the value of partial results with ongoing computational effort until reaching a final answer of maximal value [21,46,47]. We shall assume that we have access to one or more flexible algorithms, each with the ability to generate partial results $\pi(I)$ that may provide value to a system or person before a final, precise answer is reached.

4.1. Expected value of computation

A system applies a flexible strategy S to refine an initial problem instance I or previously computed *partial result*, $\pi(I)$, into a new result, terminating on a final, or *complete result*, $\phi(I)$.

In the general case, reasoning systems are uncertain about the value associated with future computation. Thus, we consider a probability distribution over results achieved with deliberation with a problem solving strategy, conditioned on the previous partial result, the computational procedure, and the allocated resources,

$$S(\pi(I), t) \rightarrow p\big(\pi'(I) \mid \pi(I), S, t\big). \tag{5}$$

Some research on flexible computation strategies has explored the application of decision theory to reason about the expected value of allocating resources to refine partial results in different settings. These efforts typically assume a utility model for assigning value to the results and costs for allocated resources [6,48].

A key construct used in studies of deliberation of flexible procedures is the *expected value of computation* (EVC) [47,48,51,61].

Definition 4. The expected value of computation (EVC) is the change in the net expected value of a system's behavior with the refinement of one or more results by computational procedures with the allocation of computational resources, t.

To compute the EVC, we consider the probability distribution over outcomes of computation and the costs of computation. If we assume that cost is a deterministic function

of computation time, and that it is separable from the value of computation, the EVC for refining a single result is,

$$\mathrm{EVC}\big(\pi(I), S_i, t\big) = \int_j p\big(\pi'_j(I) \mid \pi(I), S_i, t\big) u_o\big(\pi'_j(I)\big) - u_o\big(\pi(I)\big) - \mathrm{Cost}(t), \quad (6)$$

where $u_o(\pi(I))$ is the value of a previously computed partial result $\pi(I)$. The value of the result $u_o(\pi'(I))$ without regard to the cost incurred in its computation has been referred to as the *object-level value* of result $\pi'(I)$.

4.2. Expected value of precomputation

We now turn to the value of precomputation. The *expected value of precomputation* (EVP) is the *expected* increase in the expected utility of a system's performance in a future period with the allocation of precomputation resources. We can view EVP as the expected EVC associated with the precomputation of one or more potential future instances under uncertainty.

Definition 5. The expected value of precomputation (EVP) is the expected EVC associated with precomputing one or more potential future problem instances I_i with procedures S_i with t_i^p of computational resources,

$$\mathrm{EVP} = \sum_i p(I_i \mid E)\mathrm{EVC}\big(S_i, I_i, t_i^p\big), \quad (7)$$

where $t_i^p(I_i) \leqslant t(I_i)$ and $\sum_i t_i^p \leqslant t^a$.

We now generalize the earlier focus on minimizing the expected delay for completely solving potential future problem instances to consider the problem of allocating idle-time resources with a goal of maximizing the future expected value of the behavior of the system. We wish to allocate idle resources to different potential future problem instances so as to optimize the EVP, given uncertainty in the quantity of idle resources.

Flexible computation strategies can be characterized in terms of the *rate* at which they deliver future value with precomputation. In developing EVP policies for flexible computational procedures, it is useful to consider changes in EVP with allocations of precomputation to refine one or more partial results.

Definition 6. EVP flux, $\psi(S, I, t^p)$, of precomputing a result is the instantaneous rate at which EVP changes at the point of allocating t^p seconds of precomputation time to solving problem I with strategy S,

$$\psi\big(S_i, I_i, t_i^p\big) = \frac{\mathrm{d}\,\mathrm{EVP}(S_i, I_i, t_i^p)}{\mathrm{d}t_i^p}. \quad (8)$$

For convenience, we shall assume that the selection of a strategy or sequence of strategies S is predefined or optimized for each problem instance, and we will 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 [5,6,11,21,22,48,50,51,59]. The choice of strategy does not influence the basic results on policies for continual computation. We shall use $\psi(I,t)$ as shorthand to refer to the EVP flux associated with the strategy–instance pair after t seconds of precomputation.

We shall develop continual computation policies for sequencing precomputation effort among potential future problem instances by considering the ability of procedures to generate $\psi(I,t)$. Given information about EVP flux, we consider the total EVP of allocation policies by integrating over the EVP flux for resources allocated to each instance and summing together the EVP derived from precomputing each problem instance,

$$\text{EVP} = \sum_i \int_0^{t_i^p} \psi(I_i, t)\,\mathrm{d}t. \tag{9}$$

Given uncertainty in the amount of idle time, the overall optimization problem is finding the ideal set of allocations of the total idle time for precomputing each problem instance, I; the goal of an "anxious" automated reasoner is to allocate idle resources to strategy–problem instance pairs under consideration so as to maximize the total EVP for the idle period. In the general case, we must consider the details of the probability distribution over idle time, $p(t^p \mid E)$, and employ exact or approximate general optimization methods to choose the best set of allocations.

In lieu of employing general optimization, we seek to develop theorems analogous to Theorems 1 and 2 for prototypical classes of EVP flux associated with solving sets of future problem instances. We shall consider continual computation policies for families of partial results with refinement profiles described with utility models summarized in Fig. 1. We focus specifically on the case of problems that yield EVP flux described by linear, piecewise-linear convex, piecewise linear concave, and continuous convex and concave utility functions.

We first examine the case where we seek to harness idle-time reasoning with flexible strategies to maximize the overall increase in expected value *at the time the future*

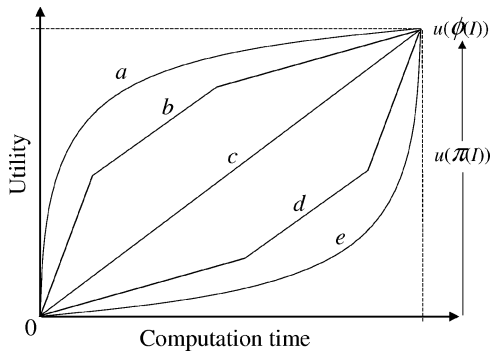

Fig. 1. Prototypical classes of utility, describing the refinement of partial results, include (a) continuous concave, (b) piecewise-linear concave, (c) linear, (d) piecewise-linear convex, and (e) continuous convex.

challenge is faced by the reasoning system. Then we will consider policies for the optimal harnessing of idle time for flexible computation that maximize the increase in expected value at the ideal time that the result should be used or that an action should be taken in the world. For these policies, we seek to maximize the expected value of the response at the ideal time that a result should be provided, based on a consideration of the costs and benefits of allocating additional real-time resources to refine a partial result.

4.3. Continual computation policies for linear utility

Let us first consider the case where computational strategies generate a constant EVP flux.

Theorem 3 (Idle resource policy for linear utility). *Given problem instances I_i that may be passed to a program in the next period, and an EVP flux $\psi(I_i, t_i^p)$ 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* EVP *flux until a final result is reached, then allocating idle time to refining the result with the next highest* EVP *flux, and so on, until the cessation of idle time or solution of all problems possible in the next period.*

By definition, the allocation of time to each instance for preselected reasoning strategies S^* applied to problem instances provide constant EVP fluxes $\psi(I_i, t_i^p) = k_i$ for each instance based on the refinement of a sequence of partial results. The total EVP is the sum of the EVP derived from precomputation of each of the future problem instances. For any amount of idle time less than the amount of time required to solve all instances under consideration, the ideal policy is to apply all resources to the instance with the highest value of k_i. Citing the same argument used in Theorem 1, any amount of time re-allocated to another instance would diminish the total EVP for cases of insufficient idle resources because it would introduce terms with suboptimal value. 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.

4.4. Flux-dominated precomputation scenarios

We can generalize Theorem 3 by introducing the property of *flux-dominated* sets of instances in precomputation settings.

Definition 7. A set of strategy–problem instance pairs is flux dominated if the set of potential future problem instances can be ordered such that the minimum EVP flux associated with precomputing any instance I_i is greater than the maximum EVP flux of solving the solution of next instance in the ordering, I_{i+1}.

Flux dominated precomputation scenarios include the case of precomputing a set of instances with distinct constant EVP fluxes. However it also covers sets of instances, including the case where problem refinement yields non-linear EVP flux in accordance with the flux-dominated property.

Theorem 4 (Idle resource policy for flux-dominated settings). *Given a set of future problem instances I_i that exhibit the flux-dominated property, the idle-time resource partition policy that maximizes the expected value at the start of the next period is to apply all resources in the order dictated by the initial* EVP *flux of precomputation. The problem with the highest* EVP *flux should be refined until a final result is reached, then the result with the next highest initial* EVP *flux should be analyzed, and so on, until the cessation of idle time or solution of all problems possible in the next period.*

We assume an ordering of instances by the minimum EVP flux they yield upon refinement. Per Eq. (9), the contribution to the total EVP of solving each instance is the integration of the EVP flux over precomputation time. We know, from the property of flux-dominated, that solving any portion of problem I_i will provide a greater EVP than the flux of solving any portion of problem I_{i+1} for all problems under consideration. The greatest contribution to EVP is provided by computing in this order. Switching any precomputation for refinement on an instance earlier in the order with one later in the order would lead to a lower over EVP under uncertain idle resources.

In the general case of nonlinear EVP flux, local decisions about EVP cannot be exploited to develop optimal policies. Identifying that problem instances under consideration are rate-dominated, allows us to invoke Theorem 4 to drive continual computation even for cases of nonlinear EVP flux. As an example, consider scenarios where EVP is described by piecewise-linear convex or continuous convex utility functions, as displayed in Figs. 1(d) and (e). With convex utility EVP is everywhere increasing and shows increasing rates of flux with the progression of problem solving. Policies based on these refinement models are typically sensitive to the probability distribution over idle time. However, if instances demonstrating such increasing EVP flux with computation also are consistent with the flux-dominated property, we can invoke an ideal continual computation policy based on a sequencing of instances by EVP flux.

4.5. Ideal policies for decreasing-returns scenarios

We now consider continual computation policies where the rate of refinement of solutions to problems slows with increasing computational activity. We refer to such scenarios of decreasing returns as concave utility.

4.5.1. Policies for piecewise-linear concave utility

Let us first consider policies for sets of instance-strategy pairs that generate EVP flux that are described by piecewise-linear concave functions, represented in Fig. 1(b). These models represent the case where EVP increases with computation but successive segments of computation are associated with decreasing EVP flux. Ideal policies for continual computation with such a utility model can be derived by generalizing the results for the case of constant EVP flux. Rather than associate problem instances with constant levels of EVP flux, we consider the flux associated with successive components of the solution of problems, and consider ideal strategies for solving these components.

Theorem 5 (Idle resource policy for piecewise-linear concave utility). *Given problem instances I_i that may be observed in the next period and* EVP *flux $\psi(I_i, t_i^p)$, described by a piecewise-linear concave model, the resource allocation policy that maximizes the expected value at the start of the next period is to allocate resources to the problem segment that has the greatest* EVP *flux of all available next problem segments, and to continue to refine the selected instance until the segment is completed, and then to allocate resources to the segment across all instances with the next highest* EVP *flux, and to continue this process until all segments of all problems are solved or the cessation of idle time.*

We generalize Theorem 3, centering on the ideal allocation of resources for problems showing constant flux, by considering distinct regions of refinement of problems under consideration, each labeled with a constant EVP flux as represented by the slope of segments of the piecewise-linear concave utility models. Given decreasing returns, we know that computation associated with earlier problem segments of instances will necessarily have higher expected value flux than refinements associated with later segments of the instances. Decreasing returns also tells us that an unsolved problem segment that currently has a larger EVP flux than all other next available segments yields greater EVP flux than can be provided by all other segments derived from any of the problems under consideration. The arguments in Theorem 3 coupled with everywhere decreasing returns constrain the policy with the greatest expected utility to be to continue to apply all precomputation resources to the problem with the greatest EVP flux, until reaching the end of the current EVP flux segment, then checking the next available segments of refinement for all available problems and again selecting the problem associated with the segment offering the highest expected flux. For the general case of having less precomputation resources than the total usable idle time, selecting any other segment except the one with the greatest EVP flux leads to a diminishment of the overall expected value at the start of the next period.

The decreasing returns property allows us to construct ideal policies by continuing to inspect only the next available problem segments for each unsolved problem to identify the best sequence of segments. The policy for maximizing the expected value in the next period is to continue to apply all resources to solving portions of problems in a sequence dictated by the EVP flux of next available problem segments, allowing for jumps across problems at the completion of each segment.

Fig. 2 captures the method for constructing an ideal sequence of segments from a set of problems whose refinements are captured by piecewise-linear concave utility models. We first transform the rate of refinement associated with computing each problem instance into *expected* fluxes for solving future problems given uncertainty in their occurrence. Then, an ideal policy is composed by continuing to expend all resources to solve the problem segment drawn from all instances that delivers the greatest instantaneous flux. In the case represented in Fig. 2, we portray a system considering three future problems, I_1, I_2, and I_3 under uncertainty. The rates of refinement under certainty are transformed into EVP fluxes and a sequence is constructed by continuing to select the problem segments with the next greatest EVP flux. In this case, we select segment 1 of problem I_1 (segment $I_{1,1}$), then segment 1 of problem I_2 (segment $I_{2,1}$), then segment 2 of problem I_2 (segment $I_{2,2}$), then segment 2 of problem I_1 (segment $I_{1,2}$), and so on,

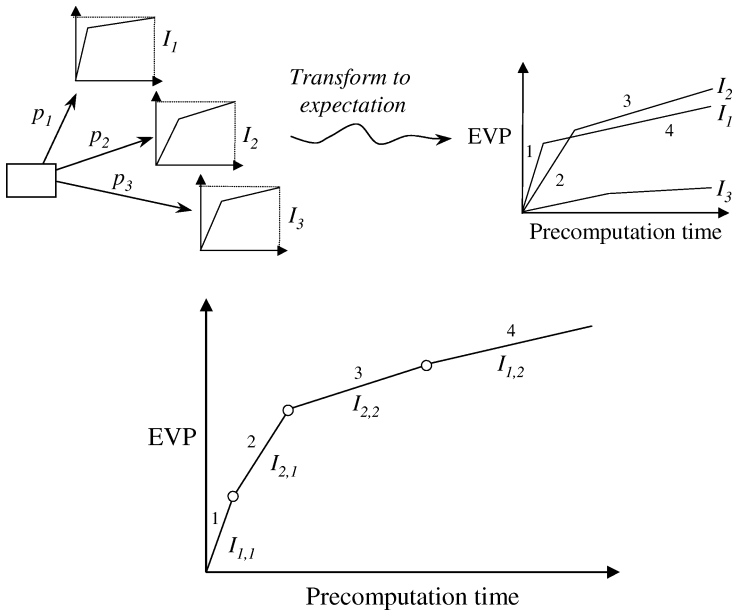

Fig. 2. EVP analysis for the case of piecewise-linear concave utility. We map the rate at which value is generated with the solution of problems under certainty to expected precomputation flux (top) and compose a precomputation policy by piecing together a sequence of segments in order of the EVP flux (bottom).

until all three problem instances under consideration are completely solved or idle time ends.

4.5.2. Policies for continuous concave utility

We can generalize Theorem 2 to the case of continuous concave utility models by taking the size of segments in the piecewise-linear models to zero in the limit.

Theorem 6 (Idle resource policy for continuous concave utility). *Given instance–strategy pairs being considered for precomputation, each showing an instantaneous change in* EVP *flux with precomputation,* $\mathrm{d}\psi(I_i, t_i^p)/\mathrm{d}t < 0$*, the resource allocation policy that maximizes the expected value at the start of the next period is to continue to shift resources to the problem with the maximal* EVP *flux, until all of the problems under consideration are solved or until the cessation of idle time.*

Given concave utility models, flux is positive but decreasing with the refinement of instances. Thus, we know that earlier precomputation provides more EVP than later refinements for any instance. The contribution to the total EVP of solving each instance is the integral of the EVP flux over precomputation time. Per Eq. (9), the policy of continuing to shift to the problem with the largest EVP flux leads to the maximum integrated EVP, maximizing the total summed EVP across instances for any amount of computation time. Changing the ordering of precomputation in any way would lead to a lower expected EVP

for situations where there is uncertainty about having sufficient precomputation time to solve all instances completely.

We can alternatively view this result as generalizing the arguments of Theorem 5 by taking the length of piecewise-linear segments to zero in the limit. The policy of continuing to select refinements with the greatest flux from all of the next available refinements is optimal even as we squeeze the size of segments down to zero.

In real-world computing, we must allocate some finite amount of computational resources to instance refinement. For EVP described by continuous concave utility functions, 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 EVP flux for that small quantity of expenditure. Because each instance has an EVP flux that is monotonically decreasing with allocation of resources, the greatest currently available flux must be greater than the future EVP flux associated with this or any other instance. Thus, guiding allocation by the local mean EVP flux leads to ideal expected overall EVP.

As portrayed in Fig. 3 we generalize the case of piecewise-linear concave utility, by mapping the instantaneous refinement with solving each instance for the deterministic situation associated to the expected flux for solving future problems under uncertainty, and allocate resources to problems in a sequence such that the expected flux is always maximal.

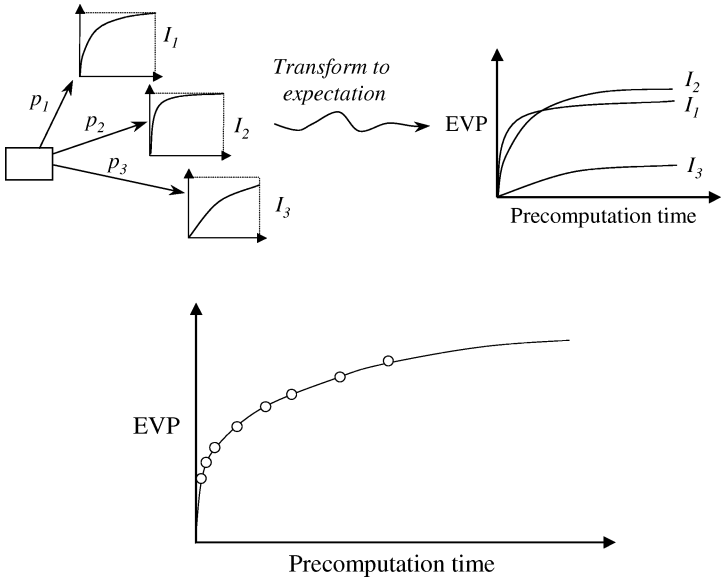

Fig. 3. EVP analysis for case of continuous concave utility. We map the rate of refinement for solving problems for solving problems to EVP flux (top) and build a policy by piecing together a sequence of segments in order of expected flux (bottom) with additional precomputation.

4.6. Approximations for general nonlinear utility

Beyond the identified utility models and properties, we cannot generally employ local EVP analysis to derive a globally optimal solution without considering multiple sequences of precomputation and the probability distribution over idle time. Thus, for the general case of nonlinear EVP under uncertain idle time, we are forced to perform more general optimization to seek policies of maximum value. In lieu of such optimizations, we can attempt to use approximate, greedy allocation strategies.

In a basic myopic approach, we consider the best next allocation of small amounts of idle time, Δt_p, assuming that idle time ends at the end of the allocation. We consider the EVP flux for an instance to be constant for small Δt^p regions and harken back to Theorem 3, substituting the mean flux during the time slice for the constant flux. Thus, at each turn, we allocate all of the time to the instance with the greatest *mean* EVP flux over this time, and continue to apply this greedy procedure as more time is available.

In an another approximation, we partition idle time into a sequence of time periods and summarize the EVP flux of instances that do not show constant or concave utility in terms of mean fluxes over progressively larger spans of time. We compose an approximate piecewise-linear utility function by substituting the mean flux of the given utility function within each period for the detailed flux. If adjacent segments of the evolving piecewise-linear functions show convexity, we recursively merge the segments and consider instead the mean flux across the merged period. We continue this process until the approximate utility function is linear or piecewise-linear concave. Give a set of instances with concave or linear models, we employ composition methods described in Section 6 to generate an approximate continual-computation policy.

5. Considering future real-time deliberation

So far, we have focused on EVP for flexible procedures targeted at the optimization of the expected value of the system at the time a new challenge is received. A system employing a flexible computation strategy should not necessarily yield its result or act immediately at the end of idle time. It may be valuable to perform additional real-time computation. We now consider policies that fold into consideration the ideal time to continue to compute a solution after a problem is encountered in real time.

Let us first examine scenarios in the absence of precomputation. Consider situations of time-critical computation captured by the graphs in Fig. 4. We refer to this class of problem as *concave value, constant cost scenarios* [50].

Definition 8. Concave value, constant cost scenarios, are real-time problems where the allocation of computational effort to the solution of problem instances provides continuous monotonically increasing values of partial results with decreasing marginal returns and the cost of reasoning increases at a constant rate with real-time delays until computation halts or a final result is generated.

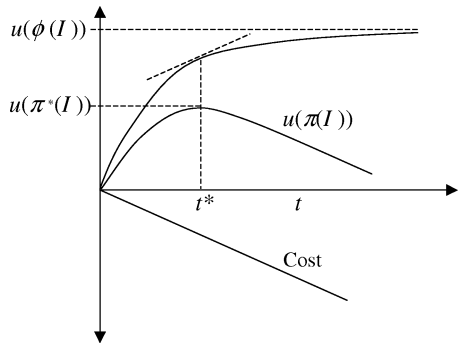

Fig. 4. Graphical analysis of ideal halting time for refining an instance with partial results showing continuous concave utility in light of a constant rate of cost. Cost is depicted on a negative scale for clarity (from [50]).

Strategies providing continuous concave refinement in the context of constant rate of cost has been explored by the community of researchers interested in time-critical reasoning. It is straightforward to identify the ideal quantity of real-time deliberation for these problems [50].

Theorem 7 (Ideal extent for concave value, constant cost problems). *In concave value, constant cost scenarios, a result should be rendered or action should be taken immediately in the world if the* EVC *at the outset of a challenge is nonpositive; otherwise, real-time computation should continue until the rate of refinement is equal to the cost of computation.*

It is worthwhile to continue computing if the EVC is positive. The instantaneous refinement of real-time computation is the difference between the rate at which the total benefit and costs of delay change with additional refinement of a result. If the cost of delay following a challenge is greater than the initial real-time refinement when a challenge occurs, the EVC of any computation is negative and, by definition of decreasing returns, will only become further negative with additional real-time computation. Thus, action should be taken immediately with the result available the current level of refinement. On the other hand, if the initial rate of refinement is greater than the cost, the EVC will be positive so it is worthwhile to further refine the result with real-time computation, even in the face of the cost of the delay. Because the rate at which the total cost is accrued is constant and the EVC flux continues to diminish monotonically with ongoing refinement, we know that the result will either be solved completely or that, sometime before this happens, the rate at which cost is accrued will become equal before becoming greater than the EVC flux and remain greater thereafter, yielding a negative EVC. Thus, action should be taken immediately without additional refinement if the initial real-time EVC flux is less then the cost; else, computation should continue until the instance is completely refined or the rate of refinement of the result becomes equal to the cost of delay.

Fig. 4 demonstrates graphically the key ideas of ideal halting for this type of real-time problem. An instance is refined into partial results $\pi(I)$ with computation or precomputation time t, eventually reaching the final result, $\phi(I)$. A broken line, tangent

to the utility of partial results, portrays the rate of refinement at the ideal halting time t^* when $\pi^*(I)$ is reached. At this point, the rate of refinement is equal to the rate at which cost is incurred. Continuing to do real-time computation after this level would incur cost faster than gains in value.

Let us now consider continual-computation policies for scenarios of concave value, constant cost. The value flux generated by a flexible procedure operating on a result is a function of the amount of time that has been invested in refining an initial problem instance. The level of refinement attained by prior computation determines the instantaneous rate of refinement associated with additional computation.

We shall refer to the partial result generated at the ideal time, t^* indicated for halting for a particular cost function $C(t)$, strategy S, and instance I, per Theorem 7 as the *ideal real-time result*, $\pi^*(I)$, for that strategy, instance, and cost function. $\pi^*(I)$ is insensitive to the amount of precomputation applied to an instance.

Consider the case where a system will face a problem instance with probability 1.0. In this case, the value of the ideal real-time result, for the case of zero precomputation, is simply the expected utility associated with action with $\pi^*(I)$, $u(\pi^*(I))$, less the total cost accrued so far. The total cost is simply the product of the rate at which cost is accrued and the ideal halting time.

Precomputation for scenarios of continuous concave utility with constant cost can be viewed as a means of attaining *cost-free refinement* of a result. Precomputation spanning the range of refinement up to the result with a quality represented by $\pi^*(I)$ can be viewed as *removing* the influence of cost during the time allocated to precomputing the result. Thus, we can view precomputation during this time as *adding value* to the utility achieved when the result reaches $\pi^*(I)$ with an EVP flux equal to the rate at which real-time cost *would have been* accrued without precomputation.

Fig. 5(a) displays graphically the impact of precomputation on the value of the result at the ideal halting time. The revised reasoning problem is represented by the solid lines held in contrast to the broken lines, representing the situation of real-time reasoning without precomputation. As portrayed in the figure, cost only begins to influence the value of the result when real-time reasoning begins. Overall, precomputation reduces the required real-time computation and raises the ultimate value of the ideal real-time result by the total real-time cost saved. Two small vertical arrows in Fig. 5(a) represent the boost in value achieved with precomputation time allocated before reaching the ideal real-time result, $\pi^*(I)$.

We now consider the case of continuing to refine a result with precomputation *beyond* the refinement represented by the ideal real-time result. Additional precomputation applied to refining a result beyond $\pi^*(I)$ adds additional value at the cost-free rate of refinement provided by the strategy at progressively greater times after the ideal halting time. Fig. 5(b) displays graphically how refining a result with precomputation beyond the quality of the ideal real-time result adds value in accordance with the flux delivered by the strategy in these regimes. Vertical arrows show the boost to the utility of the real-time result via allocation of precomputation time to refinement in regimes before and after the ideal real-time result.

In summary, we must consider two situations for computing the EVP associated with precomputing instances for scenarios of concave value, constant cost. If the level of

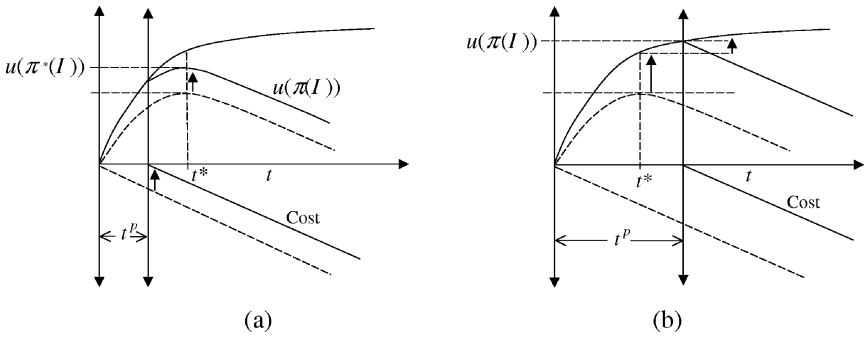

Fig. 5. (a) Graphical analysis of the influence of precomputation on ideal halting time and ultimate value of a result for the regime where partial results are less refined than the ideal real-time result. (b) Analysis of value of continuing precomputation subsequent to generating an ideal real-time result.

refinement of a result reached with precomputation is less than the value represented by $\pi^*(I)$, the EVP 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 $\pi^*(I)$, the EVP is the product of the EVC flux associated with the strategy and the probability of seeing the instance.

Armed with 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, we can adapt Theorem 7 to provide a continual computation policy. We consider all instances under uncertainty and note, for each instance, whether the partial result achieved so far with precomputation falls short of the quality of $\pi^*(I)$.

Theorem 8 (Policy for concave value, constant cost settings). *For scenarios characterized by continuous concave refinement with constant cost, it is ideal to allocate idle 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 $\pi^*(I)$ and the product of the rate of refinement and the probability of the instance when refinement of the result is greater than $\pi^*(I)$.*

6. Composing policies for mixtures of models

So far, we have focused on distinct policies for families of problem solving grouped by prototypical utility profiles. We can build composite continual-computation policies for instance–strategy pairs that span different families, including scenarios including both all-or-nothing computation and flexible procedures. Let us first cast policies for all-or-nothing computation, as described in Sections 2 and 3, in terms of EVP flux.

Precomputation scenarios allow us to transform traditional all-or-nothing computation into problems that show an incremental return for the investment of offline resources. For example, we can easily transform the continual-computation policies developed for

minimizing expected costs for all-or-nothing problems into EVP flux problems. We map the minimization of total expected run time for all-or-nothing algorithms to an EVP-flux viewpoint by assessing a utility model for differing real-time delays required to generate a final result. Utility models for delay that provide linear or monotonically decreasing increases in expected value with precomputation time can be mapped to the appropriate continual-computation policies described in this section. By invoking utility models and substituting EVP fluxes for the reduction of expected latency or cost, Theorems 1 and 2 become identical to Theorem 3.

We can compose optimal continual-computation policies for scenarios where local decisions guided by EVP flux are consistent with globally optimal EVP. Such a condition exists when can partition all problem instances into prototypical classes of precomputation problems.

First, we seek to identify a set of instance–strategy pairs representing a flux-dominated precomputation subproblem where the instance with the smallest EVP flux of the set of instances in the subproblem dominates the EVP flux provided by all instances in other classes. If such a subset exists, and all of the instances in the other classes provide EVP flux described by either constant or concave utility functions, we can use local decisions to develop a globally optimal continual-computation policy.

Theorem 9 (Composition of continual computation policies). *Given a set of potential future instance–strategy pairs that can be decomposed into a flux-dominated subset of instances, and a complement of instances that show constant or decreasing flux with refinement, the ideal allocation for continual computation is to first compute the flux-dominated instances in the order of initial* EVP *flux, followed by precomputing portions of instances that are available for refinement as directed locally by the maximal* EVP *flux provided by portions of instances, continuing until completing the refinement of all of the instances or the end of idle time.*

The composition policy follows from the earlier results. We know, by definition of flux-dominated, that we will derive the most EVP by solving these problems in the order of their initial or minimal EVP flux. After such instances are precomputed, we know that ordering computation for constant, piecewise-linear concave, or continuous concave models by EVP flux leads to an ideal EVP overall EVP for the complement of the instances. Thus, the sum of the EVP from the flux-dominated and the other instances is maximized. For instances in the flux-dominated subset and instances represented by linear utility and linear segments of piecewise-linear utility models, we can continue to precompute for an entire instance or segment without performing checks on EVP flux. However, for a set of instances providing concave utility with refinement, shifts in attention among multiple problem instances may be indicated as idle resources are expended per the maximal available EVP flux. Such consideration requires either ongoing estimation of the EVP flux available from multiple problems, or analysis (potentially offline) of the ideal sequence of precomputation identified by considering the EVP flux for functional descriptions of the utility models.

7. Continual computation over multiple periods

We have discussed policies for allocating available resources to enhancing the efficiency of problem solving in the *next period* based on a current probability distribution. In the general case, policies for directing precomputation over multiple periods must consider probabilistic relationships among instances and the probabilities of alternate sequences of instances over time.

If we assume independence among problem instances, we can employ continual computation, in conjunction with the caching of results, to maximize the response of a system over multiple periods. Given the assumption of probabilistically independent problem instances, a system can continue to perform ideal continual computation according to the policies described in Sections 2 and 3 over multiple periods. In a world abiding by such assumptions, local decisions optimize precomputation over multiple periods.

Consider the situation where continual computation has just been broken by the arrival of a specific problem instance. When idle time is broken, the results of the recent precomputation remain cached in memory. In the next idle period, we continue to refine results via precomputation, starting from the cached partial results and real-time computed result. For an unchanged probability distribution over problem instances, the overall two-period problem is equivalent to the single period continual-computation problem, extended with idle time derived from two adjacent idle periods, and with the additional value achieved through real-time computation applied to the specific instance that occurred between the two periods. We can perform the same analysis for n periods of idle time.

Now let us turn to situations where there are probabilistic dependencies among problem instances. In this situation, a system can construct a tree of future precomputation problems and can consider performing precomputation of instances at varying depths in the tree with current idle resources. Ongoing updates of the probability distributions over future problems and of continual computation policies can be made as instances and evidence are observed over time.

Deliberating about the expected value of continual computation for instances in the next period, versus a more distant period of problem solving, must take into consideration the probability distribution over idle time available in future periods. The availability of idle time in a future precomputation problem determines the value of allocating currently available idle resources to the future periods. The future distribution over idle time may be a function of the probability distribution over problems that will be encountered in advance of a future idle period under consideration.

Let us use EVP*($p(t^a \mid E)$) to refer to the ideal expected value of precomputation derived by following an optimal continual computation policy, given a probability distribution over idle time in a future period. The ideal EVP is identified by following the precomputation sequencing specified by a continual computation policy. For example, for a set of instances that each provide some constant EVP flux, we consider the precomputation sequence dictated by the continual computation policy for constant EVP flux, yielding

$$\mathrm{EVP}^*\big(p(t^a \mid E)\big) = \int_0^{t(I_1)} p\big(t^a \mid E\big)t^a \psi_1 \,\mathrm{d}t^a + \int_{t(I_1)}^{t(I_1)+t(I_2)} p\big(t^a \mid E\big)t^a \psi_2 \,\mathrm{d}t^a + \cdots + \int_{\sum_{1..n-1} t(I_i)}^{\sum_{1..n} t(I_i)} p\big(t^a \mid E\big)t^p \psi_n \,\mathrm{d}t^a. \tag{10}$$

In deliberating about the value of continuing to precompute potential problems in the next period versus in a more distant period of problem solving, we compare the EVP flux, provided by the continuing execution of the policy for the upcoming problem, with the instantaneous changes in the EVP* of future periods, achieved via allocating current precomputation resources to initiating the refinement of instances dictated by the future policy. We consider instances in the distant period as providing flux, and create compositional policies including these instances. If the current continual-computation policy is composed of instances showing linear or concave utility, we should shift to the precomputation of instances in more distant periods when

$$\frac{\mathrm{d}\,\mathrm{EVP}^*(p(t^a \mid E))}{\mathrm{d}t} > \psi\big(I_i, {t_i}^p\big), \tag{11}$$

where I_i is the instance at the current focus of attention in the continual computation policy being executed for the next problem and ${t_i}^p$ is the amount of precomputation time already allocated to the precomputation of that instance. The instantaneous change in value associated with computing instances in the next and more distant periods can diminish at different rates with the allocation of precomputation resources. Thus, continual computation policies across multiple periods can continue to shift among the refinement of instances in the different periods, just as it might within a single period of analysis.

We note that a decision making system may have time preferences about the value associated with computation and action at increasingly distant future periods. Such time preferences can be represented by a parameter that discounts the value of future results with temporal distance.

8. Allocating real-time resources to the future

So far, we have considered the allocation of available idle time for solving future challenges and have assumed that real-time resources are fully dedicated to solving current problem challenges. However, reasoning systems have the option of suspending or slowing current execution and to apply real-time resources to precomputing potential future problems.

It is worthwhile allocating resources from current to future problems when the change in the EVP of solving future problems outweighs the loss of value associated with continuing to solve the current problem. As we mentioned in Section 7, the magnitude of the boost in EVP associated with the redirection of resources to a future precomputation problem 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 all possible future problem

instances, nothing will be gained by a transfer of real-time resources to precomputation. Thus, we must consider the probability distribution over the available idle time and use this information to compute the expected value of transferring real-time resources, being directed at an existing computational challenge, to solving potential future problems.

We again consider the ideal value, EVP*, achieved by employing a continual computation policy to precompute problems in a future period, as described in Section 7. Decisions about terminating current computation to enhance precomputation influence the probability distribution over idle time. The probability distribution over idle time, $p(t^a \mid E)$, is a function of the expected time until completing or halting the current task and the probability distribution over the time a next problem instance will appear. We compare $p(t^a \mid E)$ with the probability distribution over the idle time $p(t^{a'} \mid E)$ in a world where the current problem solving were to cease before completion as dictated by the real-time refinement policy. Such a premature halt shifts the expected time required to finish the current problem to idle time. If the expected value of the gain is greater than the loss it is better to shift from computation to precomputation,

$$\mathrm{EVP}^*\big(p(t^{a'} \mid E)\big) - \mathrm{EVP}^*\big(p(t^a \mid E)\big) > u\big(\phi(I_i)\big) - u\big(\pi(I_i)\big), \tag{12}$$

where $u(\pi(I_i))$ is the current partial result achieved with real-time refinement and $u(\phi(I_i))$ is the utility of the result that would be computed by allocating all resources to real-time problem solving.

For making such decisions about allocating resources to the present versus future, a system can continue to reassess the probability distribution over idle time and compare the current expected value of precomputation versus the value of continuing to refine the current problem. Such updating and re-evaluation can lead to an interleaving of computation and precomputation. In special situations, the analysis of this probability distribution may simplified per the nature of the temporal distribution of problem instances. For example, the analysis is simplified for the case where the timing of future problem instances is described by a memoryless, Poisson distribution.

9. Considering the cost of shifting attention

As we have seen, policies for continual computation often dictate a sequence of shifts of allocation of resources among problems. Shifting the focus of precomputation attention from one instance to another may incur costs stemming from swapping partial results into and out of memory and initiating new algorithmic activity. Manipulating an extremely fine grain size for resource allocations and allocation decision making can generate significant overhead in light of such costs of switching. Thus, in real-world applications, we may wish to impose a lower-bound on the size of the minimum allocation of resources. The cost of shifting attention can influence decisions about shifting to problems with greater EVP flux. However, continual computation policies can be modified to take such costs of shifting analysis into consideration.

When an EVP analysis indicates that another instance will deliver greater flux with additional precomputation, a system can seek to ascertain whether making an investment in shifting attention would have an overall positive net value. Systems should compare the

expected value of shifting precomputation to a problem instance with higher EVP flux, versus continuing with the lower flux refinement of the current problem, considering the cost of the swapping of instances and the probability distribution over the remaining idle resources. If the current problem is solved completely, there is no choice but to make an investment in moving on to refining another problem instance with the best next EVP flux. However, if the EVP flux of solving a problem at the focus of precomputational activity becomes dominated by another potential problem instance, we consider the probability distribution over idle time remaining to determine whether the expected return of the shift are greater than the costs of the shift.

As an example, consider the case of continual computation for precomputation of future problem instances that show piecewise-linear concave utility. Assume that we have just come to a bend in the utility function for the refinement of problem instance I_1 after some period of precomputation of the instance. Now, a different instance, I_2, promises the greatest EVP flux. At the current time t_o, the flux of I_1 is $\psi(I_1, t_o)$ and the flux of I_2 is $\psi(I_2, t_o)$, $\psi(I_1, t_o) < \psi(I_2, t_o)$.

We now consider the expected value of shifting precomputation from I_1 to I_2, taking a cost Cost^s as the charge for shifting attention. To compute the expected net gain in shifting from one instance to another, we consider the probability distribution over remaining idle time, t^r, given the quantity of idle time that has already been expended in the current period, t^e. We take the difference of the EVP of continuing and the EVP of switching, given uncertainty in the remaining idle time. We obtain the EVP for each strategy by integrating each EVP flux $\psi(I_i, t_o + t^r)$ with respect to the remaining idle time t^r and consider the uncertainty in the idle time. The expected value of shifting (EV Shift)to the new problem instance is,

$$\text{EV Shift} = \int_{t^r} p\left(t^r \mid t^e\right) \int_0^{t^r} \left(\psi(I_2, t_o + t) - \psi(I_1, t_o + t)\right) \mathrm{d}t \, \mathrm{d}t^r - \text{Cost}^s. \tag{13}$$

Consider the case where the analysis of shifting attention occurs over constant flux regions of both the current and the new problem instances. In such a case, we have constant fluxes, ψ_1 and ψ_2, and the expected value of shifting to the new problem instance is,

$$\text{EV Shift} = \int_{t^r} p\left(t^r \mid t^e\right)\left[t^r(\psi_2 - \psi_1)\right] - \text{Cost}^s, \tag{14}$$

which is just

$$\text{EV Shift} = \overline{t^r}(\psi_2 - \psi_1) - \text{Cost}^s. \tag{15}$$

Eq. (15) 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,

$$\overline{t^r} > \frac{\text{Cost}^s}{\psi_2 - \psi_1}. \tag{16}$$

An analogous analysis can be formulated for problem instances showing precomputation flux represented by continuous utility models and for mixes of piecewise-linear and continuous utility models. We integrate over the flux dictated by the policies for the case of shifting and not shifting and consider the probability distribution over idle time.

10. Memory, caching, and continual computation

A system continuing to perform continual computation over multiple periods has an opportunity to deliberate about retaining partial results. The value of caching partial and complete results depends on the gains associated with the future expected value of the partial results and the quantity or the cost of memory over time to store the results.

Let us consider decision making about memory usage for a fixed quantity of storage. If there is not enough memory to cache all computed instances, a system must decide which results should be retained versus discarded when more valuable results become available. A knapsack analysis can be employed to identify the ideal configuration of stored results, where cached items have a value equal to the expected utility of caching the partial result in memory, $\mathrm{EVM}(\pi(I_i))$ and an expected memory cost of the quantity of memory required to store the result, Cost^m.

$\mathrm{EVM}(\pi(I_i))$ is the difference between the expected utility of having and not having result $\pi(I_i)$ in memory, weighted by the probability of seeing the result. Approximations for EVM include taking the difference of the expected value of having and not having the result in a computationless setting, where the best result available is used in real time through the application of situation–action rules. More comprehensive analyses take into consideration the additional gains in value achieved by precomputation and real-time computation should an instance be seen for having and not having the result,

$$\mathrm{EVM}\big(\pi(I_i)\big) = p(I_i \mid E)\big(u\big(S, \pi(I_i), t\big) - u(S, I_i, t')\big), \tag{17}$$

where t and t' represent the total precomputation and real-time resources allocated by existing policies to the further refinement of $\pi(I_i)$ and I_i respectively. The additional refinement of a future instance depends on precomputation policies, the probability distribution over idle time, the inferred likelihoods of future instances, and the cache status of other instances under consideration. For an invariant probability distribution over problems, we can compute the expected additional precomputation and real-time computation that $\pi(I_i)$ and I_i would be expected to receive and the ultimate value of the cached result. Such computation can be performed as part of ongoing continual computation. In one approach, systems can defer caching decisions by allowing temporary storage for a bounded amount of time to allow appropriate utilities to come available in the normal course of continual computation.

Given assessments of the expected value and memory costs of cached results, we can optimize the total expected value of a store via a knapsack analysis. Solving the knapsack problem optimally is an NP-complete problem [24]. However, we can employ approximate greedy or semi-greedy algorithms in an attempt to maximize the overall expected value

of cached results. For example, with a *value-density* approximation, we prioritize partial results for storage in an order dictated by the ratio of $\mathrm{EVM}(\pi(I_i))$ and $\mathrm{Cost}^m(\pi(I_2))$,

$$\frac{\mathrm{EVM}(\pi(I_1))}{\mathrm{Cost}^m(\pi(I_1))} > \frac{\mathrm{EVM}(\pi(I_2))}{\mathrm{Cost}^m(\pi(I_2))} > \cdots > \frac{\mathrm{EVM}(\pi(I_n))}{\mathrm{Cost}^m(\pi(I_n))}. \tag{18}$$

The value-density approach is used in an approximation procedure to come within a factor of two of the maximal value storage policy. Limited search among subsets of elements under consideration can further refine the approximation [62]. In an ongoing approximate knapsack analysis, ongoing EVP precomputation and caching analysis continue to update a result store.

We note that the incrementality of continual computation can lead to problems with the caching of partial results. Consider the case where the storage of a partial result for an instance requires some constant memory. For initial precomputation the value density may be quite low, leading to the risk that a partial result may be continually discarded when memory is full. Such a problem can be addressed approximately by considering as an upper bound on value the expected value of complete results in making caching decisions.

11. Real-world applications

Continual computation methods hold promise for enhancing the use of a spectrum of computational procedures including optimization of resource allocation in operating systems and databases, transmission of information over limited-bandwidth networks, and automated planning and decision-making tasks. We now review sample applications of continual computation for decision-theoretic diagnostic systems and prefetching information over limited bandwidth channels.

11.1. Continual computation in diagnostic reasoning

Over the last fifteen years, researchers have made significant advances on probabilistic representations and inference machinery that can support diagnosis and troubleshooting in challenging real-world domains [8,32,34,36,66]. A significant number of interactive diagnostic systems based on probability and decision theory have been field in a variety of areas including healthcare, aerospace, and computing arenas. These systems, sometimes referred to as *normative diagnostic systems*, employ probabilistic models to compute probability distributions over states of interest, based on a set of findings. Many of these systems are designed for interactive use, allowing the user and computer to collaborate on the iterative refinement of a diagnosis through identifying useful new information to gather. Interactive systems typically employ an iterative approach to refining the diagnosis called *sequential diagnosis* [28]. The flow of control of sequential diagnosis is displayed in Fig. 6.

Sequential diagnosis invokes a cycle of analysis with phases of belief assignment and information gathering. In the belief-assignment phase of analysis, probabilities of alternate disorders are computed from consideration of the set of observations already

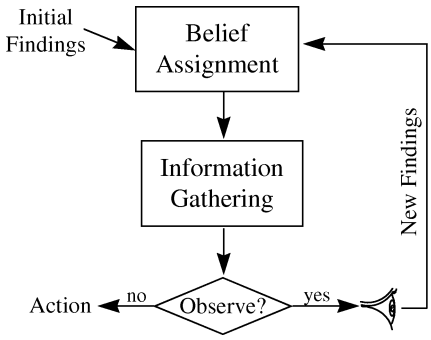

Fig. 6. Cycle of sequential diagnosis. Sequential diagnosis centers on the interleaving of phases of belief assignment and information gathering.

being considered by the system. These likelihoods are then considered by the information-gathering phase, where the best next recommendations for refining the diagnosis are computed. The recommendations generated in the information-gathering phase leads to the collection of additional evidence, which is added to the set of evidence already considered, and the system re-enters the belief-assignment phase. The cycle of belief assignment and information gathering continues until no valuable observations are available based on a cost–benefit analysis called *expected value of information* (EVI).

EVI is the difference between the value of observing new information under uncertainty, and the cost of making the observation. To compute EVI, the systems consider, for each observation, the expected value of the best action in the world for each value the observation can take on. Then, the expected utilities for each value are summed together, weighted by the probabilities of seeing the different values, should the observation be made. Given previously observed evidence, E, the systems consider the possible states that potential new observations e_x may take on after such observations are evaluated through inspection of a system in the world. Using, $e_{x,k}$ to represent the new observation that would be seen if e_x was to be observed and $\text{Cost}(e_x)$, to represent the cost of making observation e_x, we compute EVI as follows,

$$\text{EVI}(e_x, E) = \sum_k p(e_{x,k} \mid E)\left(\max_A \sum_j u(A_i, H_j)p(H_j \mid E, e_{x,k})\right) - \max_A \sum_j u(A_i, H_j)p(H_j \mid E) - \text{Cost}(e_x). \tag{19}$$

In the general case, we must consider all possible sequences of observations. This intractable computation has been avoided in practice by performing a greedy EVI analysis where systems compute or approximate the EVI for single pieces of previously unobserved evidence, under the myopic assumption that an action will be taken after the observation of one piece of evidence. The myopic EVI is used to identify the piece of evidence associated with the highest EVI.

The computation of EVI 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 [19,20]. The computation of EVI, 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. Multiple computations for each potential test or observation must be considered. Thus, even one-step lookaheads can be costly. A variety of less-expensive approximations for EVI have been explored including the selection of tests based on the minimization of entropy [3,4, 34], and the use of the statistical properties of large samples to develop value of information approximations [33].

The computation of EVI in a sequential diagnostic setting is a natural candidate for continual computation. The time required for a user to review a recommendation, make an observation or perform a test in the world, and report the result to the system can provide significant recurrent idle periods. Such idle time is an opportunity for continual computation policies to proactively cycle ahead into the uncertain future, performing computation of potential future belief updating and information-gathering analyses.

Serendipitously for continual computation, the EVI computation includes the computation of $p(e_{x,k} \mid E)$ for all unobserved evidence, representing the probability of seeing future values of observations, should the observations be made in the world. Such information, computed in the normal course of diagnostic reasoning, can be harnessed by continual computation to guide precomputation.

For the purposes of continual computation, we take as problem instances the set of potential next observations that will be made. The probability of the next observation depends in part on the way recommendations for observations are presented to the user in a diagnostic system. For example, a user may be presented with only the best next observation to make or a list of recommended observations to make ranked by EVI.

Consider the case where only the next best recommendation e_i is displayed to a user. The system has already computed probabilities $p(e_i = k \mid E)$ of seeing different states when observation e_i is made. These probabilities can be employed in a continual computation policy that considers the likelihoods of each of the k potential pieces of evidence being input to the system at the end of idle time. Continual computation can use these likelihoods to execute numerous steps of sequential diagnosis under uncertainty, precomputing and caching an ideal tree of future inferences and recommendations.

For diagnostic systems that present sorted lists of recommended new observations to users, we employ a model that provides the likelihood that a recommendation for an observation will be followed. Such a model provides the probability that a finding will be selected for evaluation based on such factors as the position of the recommended finding on a sorted list, the relative magnitude of the EVI displayed for each of the recommended findings, and the context of the diagnostic session. We compute the probability of the next evidence that will be input to the system as the product of the probability of observed state, computed by EVI, and the probability that the user will select finding y from the list, given evidence about the display list and context.

Continual computation harnessing such probabilistic information on future observations can minimize the total time of computation required in real-time by directing the available idle time to future problem instances in order of their computed probability.

11.2. Prefetching media under constrained bandwidth

Opportunities for employing principles of continual computation also arise in the communication of content in situations of scarce bandwidth for networking. The methods hold opportunity for enhancing the consumer experience of browsing content from the Internet. Currently, people often endure significant delays as information flows through the bottleneck of local modem connections. Continual-computation policies for guiding the flow of information—which we refer to as *continual networking*—can be leveraged to minimize the expected latencies associated with accessing information via such low-bandwidth communication links. Effective continual networking policies for prefecting information that may be accessed later can effectively widen the stricture on bandwidth imposed by slow connections.

Opportunities for prefetching content are underscored by typical patterns of information access displayed by people accessing information from servers which show intermittent bursts of downloading content amidst relatively long periods of idle connection time while users review the content or perform other tasks. Continual networking policies for harnessing such idle time to prefetch information into local caches have application on the client side as well as in client-server prefetching policies.

For the challenge of employing principles of continual computation to prefetching documents and media from the Internet, we consider problem instances to be the transmission of documents from servers to a local cache via the constraints of a local limited-bandwith connection. We decompose network-based content to generate partial documents and employ utility models to represent the value of having partial content available for immediate perusal.

Documents can typically be decomposed in a natural manner. A fundamental strategy for decomposing documents into partial results is simple serial truncation of content where resources are used to incrementally extend the completeness of the document. Beyond simple truncation with incremental extension, content can be decomposed through abstraction via alternate forms of summarization, and via the excision of specific classes of content. As a familiar example of excising broad classes of content, Internet browsers and servers allow users to specify whether they would like to suppress the downloading of complex graphics to speed the transmission of text associated with documents. A variety of methods that provide partial results for graphical content are already in use, centering on the flexible degradation of the resolution of images [44] employed in wavelet-based progressive transmission schemes and other schemes that manipulate generative models used in graphics such as progressive mesh simplification [38].

We have explored continual-networking policies for guiding the prefetching of documents [41]. Given a document decomposition strategy, we can employ utility models that assign value to partial content to capture the value of having immediate access to progressively larger portions of the total content of desired documents. We have pursued the use of piecewise-linear and continuous concave utility models to represent the value of having immediate access to portions of internet content as a function of the completeness of the download. These models capture the common situation where having immediate access to an initial portion of the document is most valuable and where additional content is associated with positive, but decreasing marginal increases in value with continuing down-

loading. Models of decreasing marginal utility are justified by two common features of web content. For one, the cognitive bottleneck of reviewing the initial portions of documents often allows idle resources for downloading progressively later portions of a document. Second, the typical structure of documents in combination with the typical experience of searching to see if a document has the right information, leads to the common situation where a great majority of the value of the total content is provided in the initial portions of a document.

Fig. 7(a) captures a piecewise-linear concave utility model, representing the value of having immediate access to portions of content contained in a network-based document. For this model, the value flux associated with the downloading of text is greater than that of downloading the associated graphics. Fig. 7(b) shows a model that considers a finer-grained decomposition, considering the diminishing returns associated with progressively fetching additional material in terms of serial screenfuls of text and graphics.

Let us consider prefetching based on utility functions modeled by piecewise-linear functions with decreasing rate. In practice, we allow software designers or users to specify the size of the local cache or the minimal flux required to continue prefetching. To assess value, we assume that the value of prefetching the total content for each document I is 1.0, and that the value derived from each component (e.g., successive text or graphics components) increases linearly with the portion of the component as specified by the utility model. Let us assume that the size of a component (e.g., the first screenful of text) is $B(\text{component})$ bytes and that the value derived from the component ranges linearly from 0 for no content fetched to a subutility, $u(\text{component})$, for all of that component. Given a transmission rate of $\mathcal{R}$ baud, the EVP flux associated with downloading any component associated with a segment in the piecewise-linear concave value function is:

$$\psi(\text{segment } i) = \frac{u(\text{ component } i)}{B(\text{component } i)}\, p(I \mid E)\mathcal{R}. \tag{20}$$

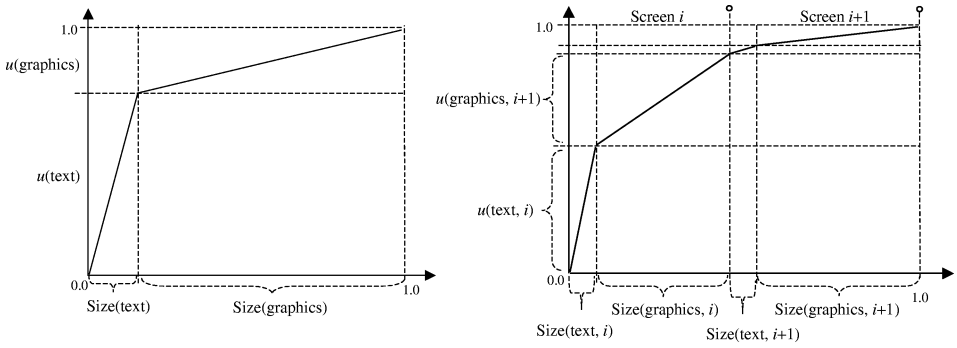

Fig. 7. (a) A piecewise-linear utility model for having immediate access to partial content where the flux of downloading text is greater than that of downloading associated graphics. (b) Finer-grained piecewise-linear model considering the diminishing returns in value of text and graphics displayed on progressively later screenfuls of content.

Prefetching policies that maximize expected utility for this utility model, described in Section 4.5, take as input quantitative or qualitative information about the likelihoods of the next documents accessed when such information is available. The prefetching policies do not require highly accurate probabilities. As we discussed earlier, the policies apply to probabilistic information available at any level of resolution. However, the policies grow in value with increases in the accuracy of inferred probabilities that a user will access documents I in a session.

We have explored several approaches to estimating the $p(I \mid E)$ for driving continual-computation policies for prefetching. Central in this task is the learning or construction of probabilistic user models that can be harnessed to infer the probabilities of next access via considering such observations as the search context, a user's browsing behavior, nature and content of documents recently visited, and the link structure of these documents. There has been growing experience with the use of Bayesian models of the goals and needs of users as they work with software applications [2,18,42]. Browsing actions with relevance to the probability of the next access include the length of time of a dwell on the current page following a new access or scroll, the dwell or access history, pattern and direction of scroll, and the mouse cursor position and patterns of motion. We have constructed such models via manual assessment techniques as well as through methods for learning probabilistic models from data. We have also employed the output of recommendation systems that may be employed during browsing. Recommendation systems include interest detection systems employing text similarity [53] and collaborative-filtering methods that identify articles of potential interest given an analysis of multiple users' activities [10,57].

Beyond expressive Bayesian models that consider user activity, interests, and link structure, we have also explored the power of using simpler statistical models of document access based on the consideration of server logs. We performed experiments from data about user activity gathered from servers to probe the promise of continual computation policies for prefetching. A set of studies centered on learning a Markov model of page accesses from anonymous user log data to generate the probability that a user will access documents in a session given the current document being reviewed.

Let us consider a representative statistical analysis that leverages information obtained from a log file of anonymous user activity accessed from the MSNBC Internet site for news

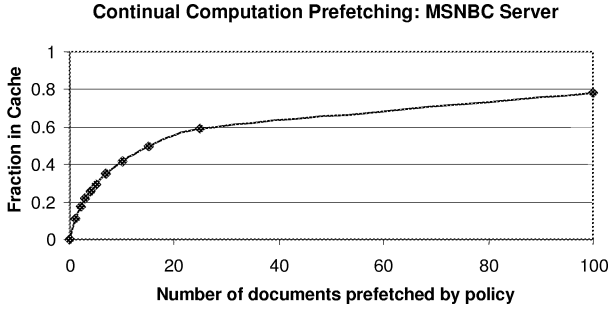

Fig. 8. Analysis of prefetching using a Markov model built from a training data from an MSNBC internet server. The graph displays the probability of content from the next document accessed being in the cache as a function of the number of documents prefetched.

and information. We processed this file to construct a Markov model of the probability, $p(I \mid \text{current document})$, for all pages in the log. The training data for this data set consists of 95,000 transitions among pages. For each page in the training set, the probability of a transition to other pages is computed by looking at all pages that had been visited next by users. The learned Markov transition probabilities provide the likelihood of users accessing the next page, and can be used to compute the probability of different sequences of pages.

Fig. 8 displays an analysis of the value of prefetching activity for the MSNBC statistical analysis on a test set of 6,000 transitions gathered from the server. If we consider documents as having identical utility models, and employ the ideal continual computation policy to fetch documents we obtain a 0.5 probability of having content in the cache for the next accessed if we prefetch content from approximately fifteen documents during recent idle time. This analysis indicates that the expected latencies are halved for this quantity of cached content. The number of documents and completeness of the content in the cache, the idle time required to generate the cache, and the size of the increase in expected value associated with such prefetching activity depends on the details of the specific utility model employed.

Beyond applications in accessing traditional web pages, the principles of continual computation can be leveraged in richer browsing experiences that provide users with the ability to navigate through high-fidelity two- and three-dimensional media. We have been exploring the use of statistical information on patterns of navigation to forecast future positions and trajectories from the current and recent history of navigational behavior—and using such forecasts to guide the proactive caching of media. Patterns of navigation vary depending on the application, and can be characterized in terms of probability distributions

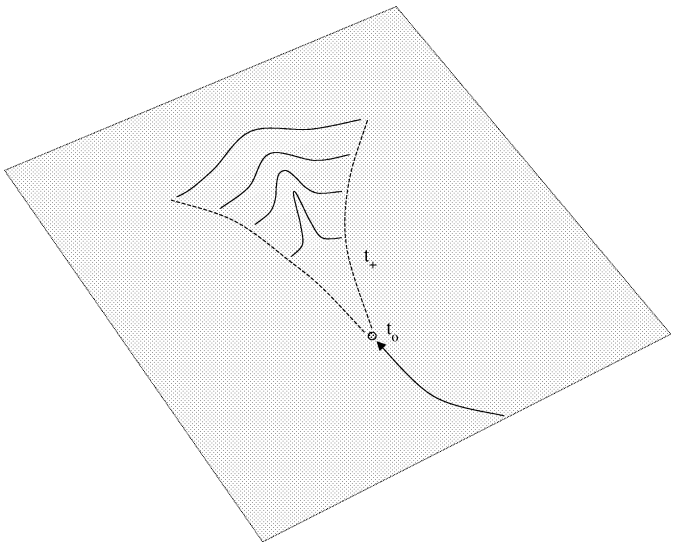

Fig. 9. Opportunities for employing continual computation include the proactive caching of media transmitted over limited bandwidth channels given forecasts of the likelihood of alternative future trajectories, based on recent navigations.

over future views conditioned on recent activity. In some applications, navigation occurs in the form of large, circling moves over broad views, followed by a pattern of drilling down for detailed views and local corrections proximal to target locations. For example, the browsing of imagery from a geographical database of satellite data involves patterns of browsing at a highly zoomed-out perspective, refining a coarse position of interest, and converging on target detailed locations of interest. Other patterns are seen during the review of content allowing walkthroughs or rotations through virtual three-dimensional spaces. As highlighted graphically in Fig. 9, probability distributions over a user's future trajectories, based on recent navigations through visual media, can be harnessed to provide guidance to continual computation policies, so as to prefetch content with bandwidth that comes available during an interactive session.

12. Related work on precomputation

Research within the Artificial Intelligence community on reasoning under bounded resources has focused largely on real-time problem solving. Nevertheless, researchers have explored the value of offline analysis and precomputation on several fronts. Prior efforts include work on real-time planning, knowledge compilation, and learning.

Greenwald and Dean [29,30] explored the challenges of harnessing offline and real-time computation in building *response planning* models for Markov decision processes under time constraints. The work pursued optimizations for more general situations of idle and real-time computation, focusing in part on issues faced with integrating reasoning systems that perform problem solving under significantly different time requirements. There have been a number of studies of offline reasoning for compilation of problem solving. Research on compilation spans challenges in decision making, theorem proving, and learning. In the realm of action under uncertainty, Heckerman, et al. investigated methods for the optimal compilation of actions given a finite quantity of memory [35]. Horsch and Poole explored the offline construction and incremental refinement of trees representing decision policies for real-time action [39]. In related work, Horvitz explored issues and opportunities with the ideal precomputation and caching of *platform results*—partial solutions to potential future real-time domain-level and metareasoning problems—to enhance the overall performance of a system [47,49]. Zilberstein and Russell have explored the offline compilation of compositions of flexible algorithms in the context of planning, developing methods for generating sequences of stretches of partial computation to optimize the value of computation [68].

There has been a rich body of work in the offline compilation of general theories to enhance the efficiency of real-time queries [13]. Representative work in this arena includes that of Kautz and Selman, centering on the development of preprocessing techniques that compile propositional logical theories into Horn theories that approximate the initial information [65]. In the work, compilation procedures translate propositional theories into a syntactic class which guarantees polynomial-time inference for future queries. In related work, Moses and Tennenholtz described offline procedures for modifying a knowledge base. They introduce restrictions to a query language to yield polynomial real-time inference [55]. In a generalization of some of the earlier work on compilation, Khardon and Roth examine the situation of a system spending offline effort in learning

about the nature of problem solving to enhance run-time competency [52]. In this work, learning procedures are employed during "grace periods" providing agents with the ability to incrementally construct a representation that increases the efficiency of handling queries expected in the environment.

Beyond artificial intelligence, precomputation methods have been explored in several computer science subdisciplines including operating systems, languages, databases, and compilers. The phrases *speculative execution*, *speculative computation*, and *optimistic computation* have been used in the systems and compilers arena to refer to procedures for precomputation centering on the execution of instructions that are not yet known to be required for a specific computation [14–16,23,25,67].

Precomputation methods in the computer systems community typically center on heuristic policies and empirical studies. Systems research on speculative execution includes the extension of computer languages and compilers with handcrafted procedures and automated techniques that create program code with the ability to compute values for multiple branches of instructions in potential windows of resource availability [12, 37,56]. The phrase *value prediction* has been used to describe a family of approaches to speculative execution centering on executing instructions before its inputs are available by guessing input values [14]. Such speculative execution of instructions can serve to break flow dependencies among instructions, enabling systems to exceed the bounds on performance based on data dependencies. Value prediction strategies typically include machinery for checking the actual values of variables downstream to verify that values were predicted correctly, and, if not, for performing a re-execution of instructions with the correct values. There is typically a cost to being wrong and a number of methods have been proposed to predicting value, including profiling the frequency of values of operands and leveraging notions of temporal locality to predict the repetition of values.

Speculative execution tasks include methods for proactively handling and making available stored data. Researchers have explored methods for loading data based on anticipated data references [58]. Moving into the realm of database research, ongoing efforts have been focused on increasing the real-time responsivity of database systems to queries by precomputing a most appropriate set of materialized views and indexes based on an analysis of the recent workload on a database system [1,17,31].

13. Toward ubiquitous continual computation

Continual computation shows promise for enhancing solutions to precomputation challenges that have been explored in several areas of computer science, including artificial intelligence, computer systems, and databases. There are numerous opportunities for modifying heuristic precomputation policies with continual computation policies, by taking advantage of coarse estimates of likelihood and the value of computation.

Beyond the effective leveraging of periods of idle time in research on the foundations of reasoning or on the fine structure of resource allocation in computer systems, continual computation can provide value in enhancing the efficiency of higher-level computational services in personal computing and Internet applications. Numerous interactive applications are characterized by intermittent bursts of computational effort

against a background of relatively long periods of idle time associated with users reflecting about action, or reviewing or inputting information. Regularities in the probabilistic structure of computer-human collaboration coupled with the relatively long periods of idle time that come with human cognition and deliberation, provide ripe opportunities for leveraging principles of continual computation.

Special formulations of continual computation might be explicitly aimed at predicting and caching responses to specific classes of problems, such as potential forthcoming computational bottlenecks. The value of employing continual computation to guide proactive planning to avoid such bottlenecks is framed by the results of research on harnessing probabilistic bottleneck diagnostic models [9]. Such models consider a set of performance indicators, software applications being used, and components of computational systems to generate probability distributions over different kinds of bottlenecks. They can also be harnessed in predicting the value of additional memory, computing, and networking resources.

Challenges with implementing continual computation include the development of efficient procedures for enumerating future instances and for estimating the value of computation and the probabilities associated with those instances. As we have seen, some applications, such as diagnostic reasoning, generate valuable probabilities of future problem instances in the course of their normal operation. However, in most cases, innovation with continual computation will hinge on the development of means for efficiently learning, accessing, or forecasting the likelihood of future problems.

There are opportunities for extending continual computation to direct precomputation at varying temporal granularities over the lifetime of systems by considering specific families of probabilistic dependency among problem instances. As another direction, rather than considering partial results as linked to specific problem instances, it is feasible that solutions to distinct problem instance might share some fundamental structure that can be precomputed and cached as partial results used in responding to multiple problem instances. Methods for noting and exploiting such shared problem structure may have significant payoffs.

Another area for future research is the study of risk preference and precomputation. We have taken an expected value perspective in developing policies. There is opportunity for introducing machinery for reasoning about uncertainties *about* probabilities of problem instances and precomputation profiles, and seeking to minimize the risks in computational decision making, potentially leveraging portfolio selection techniques that explicitly consider risk preference and that seek to identify efficient frontiers in the space of expectation and risk [54].

More fundamentally, the pursuit of continual computation can provide a window into intelligence, framing key questions and research directions on the construction of computing systems that have the ability to make the best use of constrained architectures and limited resources [47,60]. As an example, if the estimation of EVP flux for an application requires significant computation, a reasoning system may benefit by deliberating about the partition of resources among multiple activities, including the amount of effort to apply to the analysis of continual computation. Seeking to optimize the partition of resources among computation, precomputation, and deliberation about

continual computation poses a number of interesting issues with reasoning, metareasoning, and compilation under the constraints imposed by architecture and resources.

14. Summary

We presented continual-computation policies for harnessing intermittently available resources to enhance the future performance of computing systems. We focused on the identification of policies that leverage information about the likelihood of future problem instances. We considered procedures for minimizing delays and for maximizing the quality of the responses to forthcoming problems. We explored several families of utility models describing the value of partial results and derived ideal policies for guiding precomputation with local decisions for these cases. After reviewing principles, we presented illustrative applications of continual computation.

Continuing research on continual computation promises to enhance the value of computational systems and to bolster our understanding of problem solving under limited and varying resources. Future research directions highlighted by this work include developing extensions for considering policies for handling general sequences of problems over time, for considering multiple levels of temporal granularity, effectively managing memory for caching, deliberating about the likelihoods of future challenges and the value of computation, and for taking into consideration notions of risk preference for handling the case of uncertainty about probabilities and the value of computation.

Acknowledgements

I have had interesting and enjoyable conversations about continual computation with Dimitris Achlioptas, Jack Breese, Surajit Chadhuri, Jim Gray, Lloyd Greenwald, Carl Kadie, Jed Lengyel, Chris Meek, Bob Metcalfe, Natalia Moore, Mark Peot, and Harry Shum. Carl Kadie assisted with the extraction and analysis of server log data employed in prefetching studies.

References

[1] S. Agrawal, S. Chaudhuri, V. Narasayya, Automated selection of materialized views and indexes for SQL databases, in: Proc. 26th International Conference on Very Large Databases, Very Large Database Endowment, 2000.

[2] D.W. Albrecht, I. Zukerman, A.E. Nicholson, A. Bud, Towards a bayesian model for keyhole plan recognition in large domains, in: Proc. 6th International Conference on User Modeling, Sardinia, Italy, Springer, Berlin, 1997, pp. 365–376.

[3] M. Ben-Bassat, Myopic policies in sequential classification, IEEE Trans. Comput. 27 (1978) 170–178.

[4] M. Ben-Bassat, D. Teeni, Human-oriented information acquisition in sequential pattern classification: Part 1—Single membership classification, IEEE Trans. Systems Man Cybernet. 14 (1984) 131–138.

[5] M. Boddy, T. Dean, Solving time-dependent planning problems, in: Proc. IJCAI-89, Detroit, MI, AAAI/International Joint Conferences on Artificial Intelligence, 1989.

[6] M. Boddy, T. Dean, Decision-theoretic deliberation scheduling for problem solving in time-constrained environments, Artificial Intelligence 67 (2) (1994) 245–286.

[7] C. Boutilier, R. Dearden, Using abstractions for decision-theoretic planning with time constraints, in: Proc. AAAI-94, Seattle, WA, American Association for Artificial Intelligence, AAAI Press, Menlo Park, CA, 1994, pp. 1016–1022.
[8] J. Breese, E. Horvitz, M. Peot, R. Gay, G. Quentin, Automated decision-analytic diagnosis of thermal performance in gas turbines, in: Proc. Turbine Expo 92, American Society of Mechanical Engineers, 1992.
[9] J.S. Breese, R. Blake, Automating computer bottleneck detection with belief nets, in: Proc. 11th Conference on Uncertainty in Artificial Intelligence, Montreal, Que., Association for Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1995.
[10] J.S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: Proc. 14th Conference on Uncertainty in Artificial Intelligence, Madison, WI, Association for Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, CA, 1998, pp. 43–52.
[11] J.S. Breese, E.J. Horvitz, Ideal reformulation of belief networks, in: Proc. 6th Conference on Uncertainty in Artificial Intelligence, Cambridge, MA, Association for Uncertainty in Artificial Intelligence, Mountain View, CA, 1990, pp. 64–72.
[12] F.W. Burton, Controlling speculative computation in a parallel functional programming language, in: Proc. 5th International Conference on Distributed Computing Systems, 1985, pp. 453–458.
[13] M. Cadoli, F.M. Donini, A survey on knowledge compilation, AI Communications—The European Journal for Artificial Intelligence 10 (1997) 137–150.
[14] B. Calder, P. Feller, A. Eustace, Value profiling, in: Proc. 30th International Symposium on Microarchitecture, 1997, pp. 270–280.
[15] D. Callahan, K. Kennedy, A. Porterfield, Software prefetching, in: Proc. 4th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Vol. 26 (4), ACM Press, New York, 1991, pp. 40–52.
[16] F. Chang, G. Gibson, I. Automatic, Hint generation through speculative execution, in: Proc. 3rd USENIX Symposium on Operating Systems Design and Implementation (OSDI'99), 1999, pp. 1–14.
[17] S. Chaudhuri, V. Narasayya, An efficient cost-driven index selection tool for Microsoft sql server, in: Proc. 23rd International Conference on Very Large Databases, Very Large Database Endowment, 1997.
[18] C. Conati, A.S. Gertner, K. VanLehn, M.J. Druzdzel, Online student modeling for coached problem solving using bayesian networks, in: Proc. 6th International Conference on User Modeling, Sardinia, Italy, Springer, Berlin, 1997, pp. 231–242.
[19] G. Cooper, The computational complexity of Bayesian inference using Bayesian belief networks, Artificial Intelligence 42 (2) (1990) 393–405.
[20] P. Dagum, M. Luby, Approximating probabilistic inference in Bayesian networks is NP-hard, Artificial Intelligence 60 (1) (1993) 141–153.
[21] T. Dean, M. Boddy, An analysis of time-dependent planning, in: Proc. AAAI-88, St. Paul, MN, American Association for Artificial Intelligence, 1988, pp. 49–54.
[22] T.L. Dean, M.P. Wellman, Planning and control, in: Temporally Flexible Inference, Chapter 8.3, Morgan Kaufmann, San Francisco, CA, 1991, pp. 353–363.
[23] D. DeGroot, Throttling speculative computation: Issues and problems, Parallel Computing (1991) 19–37.
[24] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, New York, 1979.
[25] K. Ghosh, R.M. Fujimoto, K. Schwan, A testbed for optimistic execution of real-time simulations, in: Proc. IEEE Workshop on Parallel and Distributed Real-Time Systems, IEEE Computer Society, 1993.
[26] I.J. Good, A five-year plan for automatic chess, in: E. Dale, D. Michie (Eds.), Machine Intelligence, Vol. 2, Oliver & Boyd, Edinburgh, 1968, pp. 89–118.
[27] I.J. Good, Dynamic probability, computer chess, and the measurement of knowledge, in: E.W. Elcock, D. Michie (Eds.), Machine Intelligence, Vol. 8, Wiley, New York, 1977, pp. 139–150.
[28] G.A. Gorry, G.O. Barnett, Experience with a model of sequential diagnosis, Computers and Biomedical Research 1 (1968) 490–507.
[29] L. Greenwald, Analysis and design of on-line decision-making solutions for time-critical planning and scheduling under uncertainty, Ph.D. Thesis, Department of Computer Science, Brown University, Providence, RI, 1996.
[30] L. Greenwald, T. Dean, Anticipating computational demands when solving time-critical decision-making problems, in: Proc. 1st Workshop on the Algorithmic Foundations of Robotics, A.K. Peters, 1994.

[31] H. Gupta, V. Harinarayan, A. Rajaraman, J. Ullman, Index selection for olap, in: Proc. 13th International Conference on Data Engineering, IEEE Computer Society, 1999, pp. 208–219.
[32] D. Heckerman, J. Breese, K. Rommelse, Decision-theoretic troubleshooting, Comm. ACM 38 (3) (1995) 49–57.
[33] D. Heckerman, E. Horvitz, B. Middleton, A nonmyopic approximation for value of information, in: Proc. 7th Conference on Uncertainty in Artificial Intelligence, Los Angeles, CA, Association for Uncertainty in Artificial Intelligence, 1991.
[34] D. Heckerman, E. Horvitz, B. Nathwani, Toward normative expert systems: Part I. The Pathfinder project, Methods of Information in Medicine 31 (1992) 90–105.
[35] D.E. Heckerman, J.S. Breese, E.J. Horvitz, The compilation of decision models, in: Proc. 5th Conference on Uncertainty in Artificial Intelligence, Windsor, Ont., Association for Uncertainty in Artificial Intelligence, Mountain View, CA, 1989, pp. 162–173.
[36] M. Henrion, J.S. Breese, E.J. Horvitz, Decision analysis and expert systems, AI Magazine 12 (1992) 64–91.
[37] U. Holtmann, R. Ernst, Experiments with low-level speculative computation based on multiple branch prediction, IEEE Trans. VLSI Systems 1 (3) (1993) 262–267.
[38] H. Hoppe, Progressive meshes, in: Proc. SIGGRAPH-96, SIGGRAPH, 1996, pp. 99–108.
[39] M. Horsch, D. Poole, Flexible policy construction by information refinement, in: Proc. 12th Conference on Uncertainty in Artificial Intelligence, Portland, OR, Morgan Kaufmann, San Francisco, CA, 1996, pp. 315–324.
[40] E. Horvitz, Models of continual computation, in: Proc. AAAI-97, Providence, RI, American Association for Artificial Intelligence, AAAI Press, Cambridge, MA, 1997, pp. 286–293.
[41] E. Horvitz, Continual computation policies for utility-directed prefetching, in: Proc. 7th International Conference on Information and Knowledge Management, ACM Press, New York, 1998, pp. 175–184.
[42] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, K. Rommelse, The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users, in: Proc. 14th Conference on Uncertainty in Artificial Intelligence, Madison, WI, Morgan Kaufmann, San Francisco, CA, 1998, pp. 256–265.
[43] E. Horvitz, A. Klein, Reasoning, metareasoning, and mathematical truth: Studies of theorem proving under limited resources, in: Proc. 11th Conference on Uncertainty in Artificial Intelligence, Montreal, Que., Association for Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, CA, 1995, pp. 306–314.
[44] E. Horvitz, J. Lengyel, Perception, attention, and resources: A decision-theoretic approach to graphics rendering, in: Proc. 13th Conference on Uncertainty in Artificial Intelligence (UAI-97) Providence, RI, Morgan Kaufmann, San Francisco, CA, 1997, pp. 238–249.
[45] E. Horvitz, A. Seiver, Time-critical action: Representations and application, in: Proc. 13th Conference on Uncertainty in Artificial Intelligence (UAI-97) Providence, RI, Morgan Kaufmann, San Francisco, CA, 1997, pp. 250–257.
[46] E. Horvitz, S. Zilberstein, Proceedings of the Fall Symposium on Flexible Computation, Cambridge, MA, Technical Report FS-96-06, American Association for Artificial Intelligence, 1996.
[47] E.J. Horvitz, Reasoning about beliefs and actions under computational resource constraints, in: Proc. 3rd Conference on Uncertainty in Artificial Intelligence, Seattle, WA, AAAI and Association for Uncertainty in Artificial Intelligence, Mountain View, CA, 1987, pp. 429–444. Also in: L. Kanal, T. Levitt, J. Lemmer (Eds.), Uncertainty in Artificial Intelligence 3, Elsevier, Amsterdam, 1989, pp. 301–324.
[48] E.J. Horvitz, Reasoning under varying and uncertain resource constraints, in: Proc. AAAI-88 St. Paul, MN, Morgan Kaufmann, San Mateo, CA, 1988, pp. 111–116.
[49] E.J. Horvitz, Rational metareasoning and compilation for optimizing decisions under bounded resources, in: Proc. Computational Intelligence 89, Milan, Italy, ACM, New York, 1989.
[50] E.J. Horvitz, Computation and action under bounded resources, Ph.D. Thesis, Stanford University, Stanford, 1990.
[51] E.J. Horvitz, G.F. Cooper, D.E. Heckerman, Reflection and action under scarce resources: Theoretical principles and empirical study, in: Proc. IJCAI-89, Detroit, MI, International Joint Conference on Artificial Intelligence, 1989, pp. 1121–1127.
[52] R. Khardon, D. Roth, Learning to reason, in: Proc. AAAI-94, Seattle, WA, AAAI Press, 1994, pp. 682–687.
[53] H. Leiberman, An agent that assists web browsing, in: Proc. IJCAI-95, Montreal, Que., Morgan Kaufmann, San Francisco, CA, 1995.

[54] H.M. Markowitz, Portfolio Selection, Blackwell, Cambridge, MA, 1991.
[55] Y. Moses, M. Tennenholtz, Off-line reasoning for on-line efficiency, in: Proc. IJCAI-93, Chambéry, France, Morgan Kaufmann, San Francisco, CA, 1993, pp. 490–495.
[56] T. Mowry, M. Lam, A. Gupta, Design and evaluation of a compiler algorithm for prefetching, in: Proc. 5th International Conference on Architectural Support for Programming Languages and Operating Systems, 1992, pp. 62–73.
[57] P. Resnick, H. Varian, Recommender systems, Comm. ACM 40 (3) (1997) 56–58.
[58] A. Rogers, K. Li, Software support for speculative loads, in: Proc. 5th International Conference on Architectural Support for Programming Languages and Operating Systems, 1992, pp. 38–50.
[59] S. Russell, Fine-grained decision-theoretic search control, in: Proc. 6th Conference on Uncertainty in Artificial Intelligence, Cambridge, MA, Association for Uncertainty in Artificial Intelligence, Mountain View, CA, 1990.
[60] S. Russell, D. Subramanian, R. Parr, Provable bounded optimal agents, in: Proc. IJCAI-93, Chambéry, France, 1993, pp. 338–344.
[61] S. Russell, E. Wefald, Do the Right Thing, MIT Press, Cambridge, MA, 1991.
[62] S. Sahni, Approximate algorithms for the 0/1 knapsack problem, J. ACM (1975) 115–124.
[63] T. Sandholm, V. Lesser, Coalitions among rationally bounded agents, Artificial Intelligence 94 (1) (1997) 99–137.
[64] B. Selman, R.A. Brooks, T. Dean, E. Horvitz, T.M. Mitchell, N.J. Nilsson, Challenge problems for artificial intelligence, in: Proc. AAAI-96, Portland, OR, American Association for Artificial Intelligence, AAAI Press, Cambridge, MA, 1996, pp. 1340–1345.
[65] B. Selman, H. Kautz, Knowledge compilation using Horn approximations, in: Proc. AAAI-91, Anaheim, CA, AAAI Press, 1991, pp. 904–909.
[66] M. Shwe, B. Middleton, D. Heckerman, M. Henrion, E. Horvitz, H. Lehmann, G. Cooper, Probabilistic diagnosis using a reformulation of the INTERNIST-1/QMR knowledge base—II: Evaluation of diagnostic performance, Methods of Information in Medicine 30 (1991) 256–267.
[67] A. Sohn, Z. Wu, X. Jin, Parallel simulated annealing by generalized speculative computation, in: Proc. 5th IEEE Symposium on Parallel and Distributed Processing, 1993.
[68] S. Zilberstein, S.J. Russell, Optimal composition of real-time systems, Artificial Intelligence 82 (1–2) (1996) 181–213.