Utility-Based Termination of Anytime Algorithms

Tuomas W. Sandholm and Victor R. Lesser

Computer Science Department

University of Massachusetts at Amherst

CMPSCI Technical Report 94-54

July 1994

Utility-Based Termination of Anytime Algorithms*

Tuomas Sandholm and Victor Lesser †
{sandholm, lesser}@cs.umass.edu
University of Massachusetts at Amherst
Computer Science Department
Amherst, MA 01003

Abstract

This paper presents a method for automatically deriving a time- and content-dependent information value function for probabilistic information. This function describes analytically what real world value an agent can obtain by using a certain piece of information at a certain time. The general form of this function is formulated and a specific example with two-valued outputs is presented in the factory scheduling domain. The information value function forms a formal basis for decision-theoretic deliberation control, because the control decisions can be made in order to maximize a value directly derived from the agent's situation in its environment. We show how an expert agent can use another agent's communicated information value function to allocate the right amount of time to an anytime algorithm whose results the other agent will use. The particular anytime algorithm is generated from an incomplete algorithm by using prior execution statistics. This method enables a rational use of incomplete algorithms that are often effective, but suffer from not halting on every input. As an example, we present an anytime algorithm for determining 3-satisfiability (3SAT). The algorithm approximates 3SAT probabilistically by refining a satisfiability probability estimate over time. To enhance accuracy, both the initial satisfiability estimate and the performance profile of the anytime algorithm are parameterized by problem instance features. The result of each execution step of the algorithm is used to dynamically predict the results of future execution steps.

1 Introduction

Traditional AI systems work off-line. Their input is a complete problem description, and their output (acquired after some unknown, long time delay) is a complete answer. As problem solving systems scale up and longer execution times become common, many problem solving tasks become real-time [7], i.e. considerable changes in the real world take place during an agent's deliberation. The value of an agent's action in the real world may decrease with time, thus necessitating a tradeoff between deliberating to find a good action and performing some action early on. As an answer to the real-time requirements on AI,

*An early version of this paper appeared in [16]

†This research was supported by ARPA under ONR contract N00014-92-J-1698 and ONR contract N00014-92-J-1450. The content does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

some research has focused on reactive systems [2] that provide the needed responsiveness but are often unable to solve complex problems adequately. One alternative between these two extremes is to have the agent reason about its own reasoning, i.e. *meta-reason*. For example, the agent can choose to deliberate longer when the answer is not needed soon, but it can provide crude answers quickly if required by its environment. The qualitative tradeoff between time and quality is intuitive, but its operationalization is challenging. Some approaches use ad hoc meta-level control policies, but it is more grounded to use probability theory and utility-theory in the choice of base-level actions. We call the latter approach *decision-theoretic control*.

Three questions have to be answered to control reasoning: which deliberation tasks to execute, in what order, and how much time should be allocated to each one. This research can be divided according to two main criteria: interruptibility and number of reasoning tasks. In *design-to-time algorithms* (also called contract algorithms) the meta-reasoner implicitly sets the run time and expected result quality by setting the reasoning algorithm parameters before execution. Design-to-time algorithms are not guaranteed to have any answer available if interrupted before the planned time, and in the simplest case they have only one computational task. Garvey and Lesser [6] study the combination of several computational tasks (some using the results of others) into a composite design-to-time algorithm. *Anytime algorithms* (also called flexible computations) are examples of interruptible processing. Their result quality usually increases with time, and some answer is available at any time. Newton's iteration for root finding is an example of an anytime algorithm with one computational task. Horvitz [8] addresses the choice of a single anytime algorithm and the number of steps to run it in a medical diagnosis domain. Dean and Boddy [5, 1] study the sequential time allocation to multiple independent anytime algorithms in order to create a composite non-interruptible algorithm that maximizes a total response quality criterion. Their combination is optimal for a class of anytime algorithms. Zilberstein [19] analyzes the time allocation among multiple anytime algorithms that form a composite contract algorithm and presents a method by which this type of contract algorithm can (in theory) be interruptible. Russell and Wefald [14] study an agent that has to repeatedly choose between several deliberation actions and the currently highest ranked real world action. This can be viewed as a multitask anytime algorithm control problem, where time is allocated to deliberation actions in chunks, i.e. deliberation actions and real world actions are non-interruptible. The method has been used in adversary search and in single agent search.

Decision-theoretic control architectures need information value estimates. Russell and Wefald [14] estimate the value of a possible computation as the difference of the change in intrinsic utility and time cost. They compute the change in intrinsic utility as the change in the utility that the agent gets by not executing the real world action perceived best before the computation, but executing the action perceived best after it (either action assumed to occur after the computation). Horvitz [8] computes the comprehensive value of a computation as the product of a non-time-dependent object-related value and a time-dependent discount factor. As mentioned in both of these papers, the separation of the

information value function into two parts - one for taking time into account and one for other aspects - is not always possible in practice. In general, information value is a function of both the answer quality and time. These two factors can be reasonably separated only if they are independent. The problem of finding the exact form for the part that takes time into account is crucial, because algorithm choice and optimal time allocation are very sensitive to it [8].

This paper presents a method for automatically deriving a time-dependent, content-dependent information value function for probabilistic information. This function describes analytically what real world value an agent can get by using a certain piece of information at a certain time. It forms a formal basis for decision-theoretic control, because the control decisions can be made in order to maximize a value directly derived from the agents situation in its environment. Automating the generation of this function also removes the problem that a human expert that hand-crafts information value functions may be inaccurate or make errors in this nontrivial task. This paper extends the work of Howard [9], which describes a non-time-dependent information value function for non-probabilistic information.

The second main contribution of the paper is the use of this time-dependent information value function in determining the time allocation to an anytime algorithm. In this paper we study the termination criteria for algorithms for decision problems, i.e. problems with two possible answers. In such cases, the control architecture has to make a tradeoff between time and the probability that the answer (yes or no) is correct. As an example problem, we show how to determine the number of steps to run an anytime algorithm for determining 3-satisfiability (3SAT). The presented method can be used for any decision problem, assuming that performance profiles for the corresponding anytime algorithm are available.

The anytime algorithm that we will examine is constructed from an incomplete algorithm by using prior execution statistics. This method presents a new rational way of using incomplete algorithms that are often more efficient than complete ones, but suffer from not producing an answer on every input. A decision-theoretic criterion is used to terminate the incomplete algorithm. In order to increase algorithm accuracy, the performance profile is parameterized by problem instance features. Furthermore, the result of each execution step of the algorithm is used to dynamically predict the results of future execution steps.

Section 2 formulates a time- and content-dependent information value function for probabilistic information. Section 3 discusses approximate processing and shows how an incomplete decision algorithm can be used as an anytime approximate algorithm. Section 4 develops an information value based termination criterion for the algorithm. Section 5 presents an example and section 6 concludes.

2 Value of probabilistic information

Howard [9] analyzes the value of totally certain information. Here we present a related method to analyze the time- and content-dependent value of probabilistic information. This value is calculated as the difference of the real world value that the agent gets when acting

according to the new information at its received time t and the real world value that the agent gets when acting instantly without the new information. The value depends on the information content and when the information is received. The problem can be stated as follows. Let X be the set of possible outcome vectors $\vec{x}$, and let $\mathcal{F}$ be the set of probability density functions (pdf's) over X. Let I be the set of all possible problem instances. The information buyer (company) has a question, i.e. problem instance $i \in I$, and based on an estimated pdf $f_{est} \in \mathcal{F}$ of the answer $\vec{x} \in X$, the company will carry out an action sequence in the real world. The choice of an action sequence depends on f_{est} and t, but the value of the action sequence depends on the true posteriori $\vec{x}$ and when the action sequence was taken. The company has an initial pdf $f_c \in \mathcal{F}$ of the answer.

The information seller (expert) is to submit a pdf $f_e^t \in \mathcal{F}$ of the answer. The expert's anytime algorithm takes as an input the problem instance $i \in I$, and the output at time t is the pdf $f_e^t \in \mathcal{F}$. Communication time to and from the expert is assumed negligible. The question is, how much the company is willing to pay for the information f_e^t that it is going to use as f_{est} to choose an action sequence. The company's information value function

$$V_c^{theoretical}(f_e^t, f_c, t) = \sum_{\vec{x} \in X} f(\vec{x})[U(\vec{x}, f_e^t, t) - U(\vec{x}, f_c, 0)], \tag{1}$$

where f is the true a priori pdf over X. $U(\vec{x}, f_{est}, t)$ is the utility that the company gets when acting at time t (or later if it pleases) according to a given estimated pdf f_{est}, when the true posteriori answer is $\vec{x}$. The function U is constructed based on a domain specific model of the agent and its environment. Section 5 will present an example of this. In the formula above, the true pdf f is not known, so we approximate it with the best available estimator f_e^t. Note, that we use f_e^t, not f_c:

$$V_c(f_e^t, f_c, t) = \sum_{\vec{x} \in X} f_e^t(\vec{x})[U(\vec{x}, f_e^t, t) - U(\vec{x}, f_c, 0)]. \tag{2}$$

If the company's strategy is optimal (i.e. the company chooses an action sequence that maximizes expected value assuming that f_e^t is the true a priori pdf), then $V_c(f_e^t, f_c, t)$ will be positive for all f_e^t, f_c, and t. The value V_c of the information f_e^t depends on the information itself and the time when it is submitted. In classical information value theory [9], the price paid to the expert is the expected value of the information, not the real value that depends on the information content. Classical information value theory also omits time-dependence. The information value function presented here enables the expert to make a rational tradeoff between answer submission time and quality. Let V_e be the expert's value function. It is the amount that the company is willing to pay to the expert (i.e. company's information value) minus $c_{comp}(t)$, the cost that the expert incurs from allocating his computational resources to the task for t time units:

$$V_e(f_e^t, t) = V_c(f_e^t, f_c, t) - c_{comp}(t). \tag{3}$$

Let us make the assumption that for fixed f_e and f_c, $V_c(f_e, f_c, t)$ is a decreasing function of time t. This assumption is not too strong, because the company agent can always

postpone using f_e and thus never be worse off by receiving f_e early on. The fact that the expert's payment decreases with time motivates him to submit his answer as soon as he stops computing it.

In the case of non-probabilistic information, the truthfulness of the expert's answer can easily be evaluated. The information buyer just verifies the information by observing the real world ex post, and checks that the expert's answer equals the real world observation. The problem with probabilistic information is that the expert can act strategically and submit an answer that gives him higher profit than his truthful answer f_e^t would. The information buyer cannot detect that the expert's answer was biased, because the posteriori real world observation cannot be used to refute the a priori probability distribution submitted by the expert (unless $f_e^t(\vec{x}) = 0$ for the posteriori $\vec{x}$). Some reward functions (called proper scoring rules [17]), that motivate the expert to tell the truth have been developed, but they require that the expert is paid after the true posteriori $\vec{x}$ has been observed, and their applicability in the time-dependent case is unknown. If the buyer's action sequence affects the future events in a manner that disables posteriori observation of $\vec{x}$, the expert's truthfulness can be evaluated in neither the classical nor the presented method. The expert and the company may be the same real-world entity that uses the presented method for deliberation control, not for real information sales. This is one example, where the problem of strategic acting does not arise, because the information buyer and seller are cooperative. Similarly, if the same expert is used repeatedly, his long term truthfulness can perhaps be assessed and trust can be built.

In the next section we describe an anytime algorithm for decision problems, and in the section after that we show how the value function described above can be used to decide when to terminate the algorithm.

3 Approximate processing for decision problems

Lesser et al. [10] identify three ways to reduce solution quality for approximate processing in interpretation domains. One way is to *ignore some solution aspects*. Analogously, one could ignore some details of a solution to an optimization problem. This is not applicable in decision problems, because the answer is simply yes or no (Y/N). A second way to reduce solution quality is to compromise precision. This is analogous to *relaxing optimality* in optimization problems. This type of approximation is meaningless for decision problems, but it has been widely studied for $\mathcal{NP}$-complete optimization problems. Often, a fast algorithm exists that guarantees a solution within an error bound from optimum. To solve a troublesome problem in $\mathcal{NP}$, we could reduce it in polynomial time to an $\mathcal{NP}$-complete problem for which an approximation scheme of this type is known. There is a complication, though. In general, reductions do not preserve this type of approximation, i.e. the fact that the new reduced problem can be solved approximately within a certain error bound does not guarantee that the acquired solution is an approximate solution (within any reasonable error bound) to the original problem. The search for approximation preserving reductions,

called L-reductions, is an active field of research in theoretical computer science [13]. A third way of reducing solution quality for approximate processing is to *decrease certainty*. This is applicable to decision problems too. For example, our algorithm outputs the probability that the answer is Y.

We construct the approximate algorithm from an incomplete algorithm. We assume that the incomplete algorithm never halts if the real answer is N (solution does not exist), and that it may or may not halt if the true answer is Y (solution exists). The opposite case can be handled in a similar way. Let us define the following symbols: SOL^t = "Solution found by time t" and $NOSOL^t$ = "No solution found by time t". The approximate algorithm emerges when we realize that the probability of the answer being Y decreases with the number of steps that the algorithm is executed (unless, of course, the algorithm halts, which guarantees that the answer is Y). This probability ($p(Y|NOSOL^t)$) can be computed from the performance profile $p(SOL^t|Y)$ of the algorithm (the probability of finding a solution by time t, given that a solution exists). The performance profile can be constructed from prior runs of the algorithm as will be presented later.

We use 3SAT as an example decision problem due to its central role in computer science and in order to present how an incomplete algorithm can be converted to an anytime algorithm. 3SAT is the decision problem of whether a satisfying truth assignment exists for variables in a 3CNF formula. A 3CNF formula is a conjunct of clauses, where each clause is a disjunct of three literals. A literal is a negated or non-negated variable. The formula is satisfiable (corresponding to answer Y in our domain), if the variables can be assigned Boolean values so that the formula evaluates to true. For example, the formula $(v_1 \vee v_2 \vee v_3) \wedge (v_1 \vee v_3 \vee v_4) \wedge (v_1 \vee v_2 \vee v_4)$ is satisfiable by the truth assignment $v_1 = true, v_2 = true, v_3 = true, v_4 = false$ (among others). We will call the number of variables v and the number of clauses c. In the formula above, $v = 4$ and $c = 3$. The importance of 3SAT lies in the fact that it is structurally simple yet $\mathcal{NP}$-complete, i.e. any problem in $\mathcal{NP}$ can be reduced to 3SAT in polynomial time. A decision algorithm for 3SAT can (in theory) be used to solve any problem in $\mathcal{NP}$. The complexity of solving an arbitrary problem in $\mathcal{NP}$ is at most a polynomial factor greater than the complexity of the algorithm for 3SAT.

We generated 3CNF formulas using the standard method [11] for constructing hard 3SAT instances: for every clause, pick three variables randomly disallowing duplicates, and then negate each variable separately with probability 0.5. The presented estimation of satisfiability probability is based on a statistical analysis of 3SAT instances from this distribution. Therefore, the approximation algorithm is not necessarily accurate for instances from a different distribution (e.g. reduced from a different problem).

4 Terminating incomplete decision algorithms

In any decision problem, $\vec{x} \in \{[Y]^T, [N]^T\}$. Therefore, $p([Y]^T) = 1 - p([N]^T)$ completely characterizes a pdf $f \in \mathcal{F}$. Without loss of generality, we switch to a scalar notation,

and characterize f by $p(Y)$. Now we are ready to state the criterion for terminating the anytime algorithm. If the algorithm has found a definite answer (Y), the expert is motivated to submit this answer immediately, because his payoff decreases with time as mentioned in section 2. Next, the case where an answer has not been found is analyzed. Let $E^{t_{cur}}[V_e^t]$ be the expert's expectation at some current time t_{cur} of the value he will have acquired by some later time t. This value is computed as the weighted sum of the expert's possible outcomes: not having a definite answer by time t, and having a definite answer (Y) by some time step between the current time and t. Note that the expert's expectation of a fixed future time changes as the algorithm executes further. The expert should continue running the anytime algorithm as long as he expects a benefit from doing so, i.e. as long as at each reached current time step t_{cur},

$$\exists t > t_{cur} \text{ s.t. } E^{t_{cur}}[V_e^t] > V_e^{t_{cur}}, \tag{4}$$

where

$$\begin{aligned} E^{t_{cur}}[V_e^t] &= p_e(NOSOL^t|NOSOL^{t_{cur}})V_e(p_e(Y|NOSOL^t),t) + \\ &\quad \sum_{\tau=t_{cur}+1}^{t} p_e(SOL^\tau \wedge NOSOL^{\tau-1}|NOSOL^{t_{cur}})V_e(1.0,\tau) \\ &= p_e(NOSOL^t|NOSOL^{t_{cur}})V_e(p_e(Y|NOSOL^t),t) + \\ &\quad \sum_{\tau=t_{cur}+1}^{t} p_e(SOL^\tau|NOSOL^{\tau-1})p_e(NOSOL^{\tau-1}|NOSOL^{t_{cur}})V_e(1.0,\tau) \end{aligned} \tag{5}$$

and $V_e^{t_{cur}} = V_e(p_e(Y|NOSOL^{t_{cur}}), t_{cur})$. Next we describe how the quantities needed to solve these equation can be computed. Let τ_1 and τ_2 be arbitrary times s.t. $\tau_1 \leq \tau_2$. By the definition of conditional probability,

$$p_e(NOSOL^{\tau_2}|NOSOL^{\tau_1}) = \frac{p_e(NOSOL^{\tau_2} \wedge NOSOL^{\tau_1})}{p_e(NOSOL^{\tau_1})} = \frac{p_e(NOSOL^{\tau_2})}{p_e(NOSOL^{\tau_1})} \tag{6}$$

and $p_e(SOL^{\tau_2}|NOSOL^{\tau_1}) = 1 - p_e(NOSOL^{\tau_2}|NOSOL^{\tau_1})$. According to the Bayes rule, the chance that the answer is Y given that no solution has been found so far (by time t) is

$$p_e(Y|NOSOL^t) = \frac{p_e(Y)p_e(NOSOL^t|Y)}{p_e(Y)p_e(NOSOL^t|Y) + p_e(N)p_e(NOSOL^t|N)}, \tag{7}$$

where $p_e(NOSOL^t|N) = 1$ and $p_e(NOSOL^t|Y) = 1 - p_e(SOL^t|Y)$.

Solving Equation 5 for different $t_{cur} \in [0, t_{max}]$ and $t \in]t_{cur}, t_{max}]$ gives (unless the optimal number of steps to run is t_{max}) some t^* that is the smallest t_{cur} s.t. $\forall t > t_{cur}, E^{t_{cur}}[V_e^t] \leq V_e^{t_{cur}}$. So if the algorithm has not halted by time step t^*, it should be terminated at that time and the expert should submit the corresponding answer ($p_e^{t^*}(Y) = p_e(Y|NOSOL^{t^*})$) to the company. On the other hand, if the algorithm happens to halt at some time $t_{cur} \leq t^*$,

the expert should submit his answer ($p_e^{t_{cur}}(Y) = 1.0$) immediately, because now his value function $V_e(1.0, t)$ is a strictly decreasing function of time.

The presented method of trying all combinations of t_{cur} (within the possible maximum number of steps) and t (within the possible maximum number of steps, $t > t_{cur}$) before executing the anytime algorithm is quadratic in the number of possible steps. Quadratic performance is accomplished by storing the value of $\sum$ in Equation 5 while sweeping through t. Alternatively, termination can be decided in an on-line manner after each step of the anytime algorithm by a linear sweep of t in the possible range. Because the maximum number of possible steps may be very large (eg. exponential in the number of variables in 3SAT in the worst case), both methods may be impractical. Therefore, it may be appropriate to just approximate the termination method by considering termination only at some infrequent t_{cur}, thus possibly running the anytime algorithm for longer than it would have been run if the detailed termination analysis had been done. Another approximation would be to consider only some infrequent t. For example, to only consider every k:th t (always starting at $t_{cur} + k$), Equation 5 can be modified to

$$E^{t_{cur}}[V_e^t] \approx p_e(NOSOL^t|NOSOL^{t_{cur}})V_e(p_e(Y|NOSOL^t), t) + \sum_{\tau=t_{cur}+k,\ Step\ k}^{t} p_e(SOL^\tau|NOSOL^{\tau-k})p_e(NOSOL^{\tau-k}|NOSOL^{t_{cur}})V_e(1.0, \tau). \quad (8)$$

This leads to possibly terminating the anytime algorithm sooner than in the case of detailed termination analysis, because this equation gives a lower bound for the true $E^{t_{cur}}[V_e^t]$. The two approximation methods can also be combined to reduce the time to decide termination. In the following application example we will use neither approximation, but the detailed termination analysis instead.

5 Application example

As an example of the use of the information value function, we show how it can be used to terminate incomplete algorithms for decision problems, i.e. problems with two possible answers. We study the following simplified problem of possibly terminating a manufacturing project before completion in order to avoid useless costs. A company is offered 60,000 if it produces an item by time 30,000. If the project does not meet its deadline, the contract becomes void: nothing is paid, and the project need not be finished. The company's project manager knows, that the cost of the project is 1 per time unit. He also knows that the project cannot be finished before time 30,000, and he estimates that there is a 60% chance of finishing the project on time, i.e. exactly at time 30,000. He calculates the company's expected profit: $0.6(60,000 - 30,000 * 1) + 0.4(0 - 30,000 * 1) = 6,000$. As a utility maximizing risk neutral agent, he accepts the contract. Work on the project starts and the costs begin to cumulate. The last phase of the project is machining, where the key question is: can the machining be scheduled to begin at time 15,000 and to finish by time 30,000? If not, obviously the sooner (during the earlier phases) the project is terminated,

the better. Similarly one can see that if the machining can be successfully scheduled, it is worth completing the project. If the project is prematurely terminated, it cannot be restarted. Once the machining phase starts, it cannot be terminated until time 30,000.

Let us call the possible answers to this decision problem $Y =$ "Machining can be scheduled to begin at 15,000 and finish at 30,000" and $N =$ "Machining cannot be scheduled to begin at 15,000 and finish at 30,000". We assume that the project manager reduces the machining scheduling decision problem instance to 3SAT before submitting it to the expert, so analogously $Y =$ "satisfiable" and $N =$ "not satisfiable". Before submitting the problem instance to the expert, the project manager computes the time-dependent content-dependent information value function for the expert's probabilistic answer. The company's own pdf f_c is described by $p_c(Y) = 0.6$. The company's alternative utilities U are demonstrated in Figure 1. If the project is terminated prematurely, the true posteriori satisfiability cannot be observed, and $p_{est}(Y)$ cannot be evaluated for correctness.

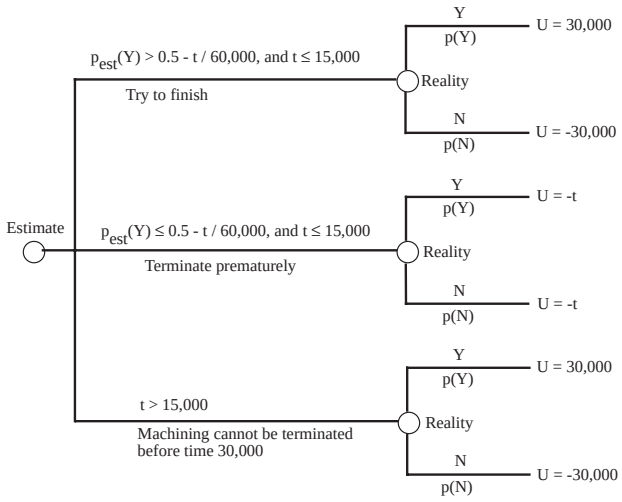

Figure 1: *The tree represents the company's optimal decisions and utility, when it is given some estimate $p_{est}(Y)$ (by itself or by an expert). The true a priori probability is $p(Y)$, and the real posteriori event is Y or N.*

By using $p_e^t(Y)$ as f_e^t, the company's information value function (Equation 2) can be written as follows:

$$\begin{aligned} V_c(p_e^t(Y), p_c(Y), t) &= \sum_{\vec{x} \in \{Y,N\}} p_e^t(\vec{x})[U(\vec{x}, p_e^t(Y), t) - U(\vec{x}, p_c(Y), 0)] \\ &= p_e^t(Y)[U(Y, p_e^t(Y), t) - U(Y, 0.6, 0)] + \\ &\quad [1 - p_e^t(Y)][U(N, p_e^t(Y), t) - U(N, 0.6, 0)], \end{aligned}$$

where the U-values can be calculated from the company agent's situation in its environment. For example, if the true answer is Y and the probability estimate of the true answer being

Y is $p^t_{est}(Y)$, the company will get a utility value (its payment minus its cost) of $60,000 - 30,000 = 30,000$ if the project is not terminated prematurely, and a utility value (cost so far) of $-t$ if the project is terminated prematurely. The project will not be prematurely terminated if it is already too late to terminate it by definition ($t > 15,000$) or the expected value of continuing is higher than that of terminating ($p^t_{est}(Y) * 60,000 > 30,000 - t$). So,

$$U(Y, p^t_e(Y), t) = \begin{cases} 60,000 - 30,000 = 30,000 & \text{if } p^t_e(Y) > \frac{30,000-t}{60,000},\ t \leq 15,000 \\ -t & \text{if } p^t_e(Y) \leq \frac{30,000-t}{60,000},\ t \leq 15,000 \\ 60,000 - 30,000 = 30,000 & \text{if } t > 15,000. \end{cases}$$

Similarly,

$$U(Y, 0.6, 0) = 60,000 - 30,000 = 30,000,$$

$$U(N, p^t_e(Y), t) = \begin{cases} 0 - 30,000 = -30,000 & \text{if } p^t_e(Y) > \frac{30,000-t}{60,000},\ t \leq 15,000 \\ -t & \text{if } p^t_e(Y) \leq \frac{30,000-t}{60,000},\ t \leq 15,000 \\ 0 - 30,000 = -30,000 & \text{if } t > 15,000, \end{cases}$$

$$U(N, 0.6, 0) = 0 - 30,000 = -30,000.$$

Now, substituting the U's in place, the company's information value function submitted to the expert becomes

$$V_c(p^t_e(Y), 0.6, t) = \begin{cases} -60,000 p^t_e(Y) - t + 30,000 & \text{if } p^t_e(Y) \leq \frac{30,000-t}{60,000},\ t \leq 15,000 \\ 0 & \text{otherwise.} \end{cases}$$

Let the expert's cost for allocating computational resources for duration t be $c_{comp}(t) = 0.5t$. The expert's value function (Equation 3) becomes

$$V_e(p^t_e(Y), t) = \begin{cases} -60,000 p^t_e(Y) - 1.5t + 30,000 & \text{if } p^t_e(Y) \leq \frac{30,000-t}{60,000},\ t \leq 15,000 \\ -0.5t & \text{otherwise.} \end{cases}$$

Figure 2 presents contour plots of the company's and the expert's value functions. In general, these function need not be piece wise linear for the method of this paper to apply. In the next section we show how this example value function can be used to terminate an anytime algorithm.

5.1 Initial solution approximation

Often, anytime algorithms have a mandatory phase, during which a rough solution for the problem is generated. This rough solution is then enhanced in an anytime refinement phase. The mandatory phase can be viewed as a non-interruptible setup phase for the actual refinement algorithm. For example in the TSP, the mandatory phase could be a greedy generation of an initial tour, and the refinement phase could be arc swapping-based local search. In 3SAT, our mandatory phase consists of generating an initial estimate of

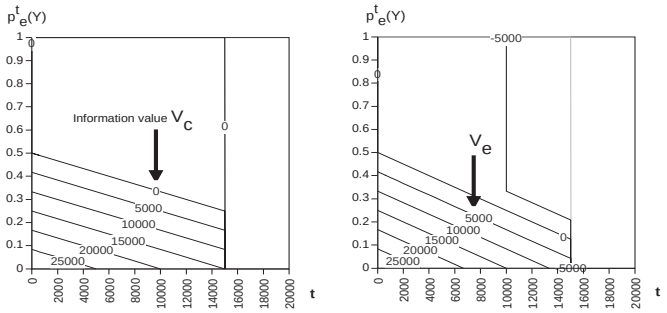

Figure 2: *Left: Contour plot of the company's information value function. Right: expert's value function.*

the satisfiability probability. For this we use three features of the problem instance as predictors: the number of variables v, the standard $\beta = \frac{c}{v}$ predictor [3, 11], and a new Δ predictor [15]. Fig. 3. shows the contours of the initial satisfiability probability. Say, that in our scheduling example of section 5 the 3SAT formula has the following features: $v = 150$, $c = 645$, (so $\beta = 4.3$), and $\Delta = 2.77$, corresponding to an arbitrary problem instance. Thus the initial satisfiability probability $p_e(Y) = 0.454$.[1]

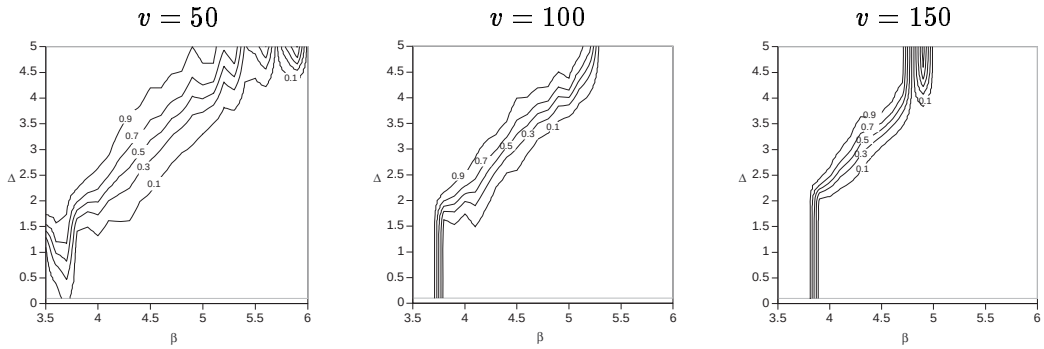

Figure 3: *Contours of the initial satisfiability probability* $p^0_e(Y)$.

Often, the mandatory phase is fast compared to the refinement phase, and in the current analysis, its time is ignored. The initial satisfiability probability can be computed in $O(c+v)$ time. One step of the refinement algorithm also takes $O(c + v)$ time.

5.2 Refinement phase

We constructed the anytime refinement algorithm from the BREAKOUT [12] satisfiability determining algorithm. BREAKOUT is incomplete, i.e. if the formula is not satisfiable, the algorithm never halts, but if the formula is satisfiable, the algorithm may halt proving satisfiability or it might not halt. The first function needed in using the anytime algorithm

[1] The way to compute this value (and the contours of figure 3) statistically is presented in [15].

is the performance profile $p_e(SOL^t|Y)$. This probability is parameterized by algorithm step t, v, β, and Δ. We gathered statistics for 50, 100, and 150 variables for $\beta = 0.1, 0.2, ..., 9.0$. At each such point, we generated 500 random formulas, determined their satisfiability using a complete algorithm similar to that of Crawford and Auton [4], and measured the Δ parameter. We ran the BREAKOUT algorithm on the satisfiable formulas and recorded its number of steps. If BREAKOUT had not found a solution by 20,000 steps, we aborted the run and recorded an unsuccessful result. Ideally one could fit a curve on the data for $p_e(SOL^t|Y)$ as a function of t, v, β, and Δ. This curve could be used for interpolation, but especially for extrapolation for large v, for which no complete satisfiability determining algorithms run in reasonable time. The problem instance from scheduling has the features $v = 150$, $\beta = 4.3$, and $\Delta = 2.77$. We construct the performance profile by looking at prior data with the same v and c. Of these runs we choose the 20 that have their Δ closest to 2.77. From the recorded number of BREAKOUT algorithm steps needed for each of these runs, we construct the performance profile $p_e(SOL^t|Y)$, Fig. 4 left.

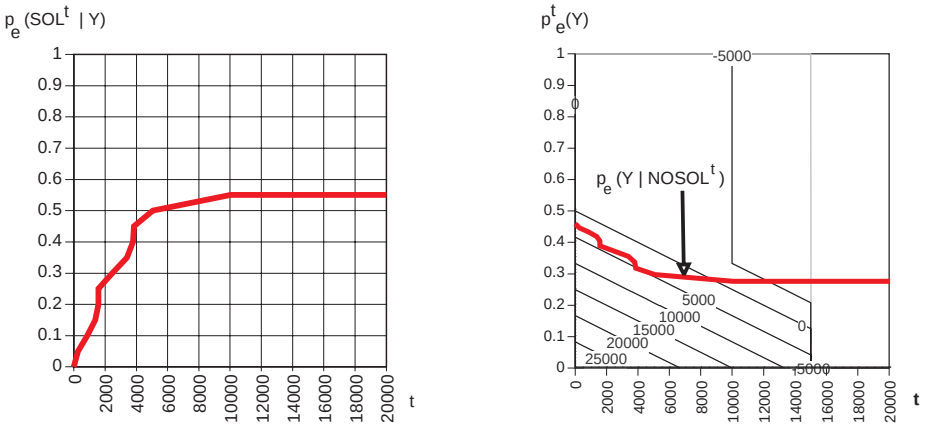

Figure 4: *Left: $p_e(SOL^t|Y)$: probability of finding a satisfying solution (halting) by algorithm step t. Right: satisfiability probability $p_e(Y|NOSOL^t)$ overlaid on the expert's value contours.*

Now we state the criterion for terminating the anytime algorithm of our example. Solving Equation 5 for different $t_{cur} \in [0, 20000]$ and $t \in]t_{cur}, 20000]$ reveals that $t^* = 1574$ is the smallest t_{cur} s.t. $\forall t > t_{cur}, E^{t_{cur}}[V_e^t] \leq V_e^{t_{cur}}$. So if the algorithm has not halted by time step 1574, it should be terminated at that time and the expert should submit the corresponding answer ($p_e^{1574}(Y) = p_e(Y|NOSOL^{1574}) = 0.384$) to the company, see Fig. 4 right. At that point the expert's value is $V_e^{1574} = V_e(0.384, 1574) = 4593$. On the other hand, if the algorithm happens to halt at some time $t_{cur} \leq 1574$, the expert should submit his answer ($p_e^{t_{cur}}(Y) = 1.0$) immediately, because now his value function $V_e(1.0, t)$ is a strictly decreasing function of time.

6 Conclusions and future research

We presented a method for automatically deriving a time- and content-dependent information value function for probabilistic information. The construction of this function was described for probabilistic answers in general, and an example was presented for decision problems via a scheduling application. The function describes analytically, what real world value an agent can obtain by using a certain piece of information at a certain time. It forms a formal basis for decision-theoretic deliberation control, because the control decisions can be made to maximize a quantity directly derived from the agent's situation in its environment. We used the information value function to help an expert agent decide how many steps of an anytime algorithm to run before submitting the output to another agent that is going to use it as a basis for its future actions.

In this paper, we presented a termination method for incomplete algorithms for decision problems in general. We used 3SAT as a canonical example. The anytime algorithm refines a 3-satisfiability probability estimate over time. The deliberation control mechanism has to trade time for certainty about the answer (satisfiable or nonsatisfiable). The anytime algorithm for 3SAT was constructed from an incomplete algorithm by using prior execution statistics. This method presents a new way of using incomplete algorithms that are often more efficient than complete ones, but suffer from not halting on every input. In order to increase algorithm accuracy, both the initial satisfiability probability estimate and the algorithm's performance profile were parameterized by problem instance features. The result of each execution step of the algorithm was used to dynamically predict the results of future execution steps.

The method can be extended to algorithm selection in the obvious way. First, count the optimum number of steps and the corresponding optimum expected information value for each alternative algorithm. Then choose the algorithm that provides the highest expected information value.

The application example of reducing a scheduling problem to 3SAT was chosen for pedagogic reasons only. To use the methods of this paper for scheduling decision problems, anytime algorithms specific to scheduling should be used. The 3SAT statistics were for a specific problem instance distribution: hard randomly generated instances. Problem instances reduced from other problems may have different statistics, and to use this method for reduced problem instances, the statistics should be collected from the corresponding problem instance distribution.

An interesting open question is the analytical formula for the initial satisfiability probability based on the mentioned problem instance features and possibly other such features. This seems to be a difficult problem: even the formula that only takes the β parameter into account is an open field of research [18]. The analytical, algorithm specific formula for the probability of finding a solution given that the formula is satisfiable is also unknown. This function is also parameterized by problem instance features. If these two formulas could be extrapolated for larger numbers of variables, our method would provide a very useful tool for controlling satisfiability determining by incomplete algorithms for problem sizes, for

which currently known complete algorithms are impractical. The method is useful for other decision problems also if the corresponding initial probability estimate and the performance profile are known and they can be extrapolated for large problem instances. In other decision problems too, the initial probability estimate and the performance profile should be parameterized by syntactic problem instance features in order to enhance accuracy. Future research includes the application of the presented method to controlling anytime algorithms in other domains and also the extension of the termination method to other problem types beyond decision problems.

Acknowledgements

We thank Shlomo Zilberstein and Neil Immerman for their helpful comments.

References

[1] Boddy, M. and Dean, T. 1989. Solving time-dependent planning problems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 979-984.

[2] Brooks, R. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14-23.

[3] Cheeseman, P., Kanefsky, B., and Taylor, W. 1991. Where the Really Hard Problems Are. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pp. 331-337.

[4] Crawford, J. and Auton, L. 1993. Experimental Results on the Crossover Point in Satisfiability Problems. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 21-27.

[5] Dean, T. and Boddy, M. 1988. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pp. 49- 54.

[6] Garvey, A. and Lesser, V. 1993. Design-to-time real-time scheduling. *IEEE Transactions on Systems, Man and Cybernetics* 23(6).

[7] Garvey, A. and Lesser, V. 1994. A Survey of Research in Deliberative Real-Time Artificial Intelligence. *Real-Time Systems*, 6, 317-347.

[8] Horvitz, E. 1989. Reasoning about Beliefs and Actions under Computational Resource Constraints. In: Uncertainty in Artificial Intelligence 3. Kanal, L., Levitt, T., and Lemmer, J. (eds.), pp. 301-324.

[9] Howard, R. 1966. Information Value Theory. *IEEE Transactions on System Science and Cybernetics*, SSC-2(1): 22-26.

[10] Lesser, V., Pavlin, J., and Durfee E. 1989. Approximate Processing in Real-Time Problem Solving. *AI Magazine* 9(1): 49-61.

[11] Mitchell, D., Selman, B., and Levesque, H. 1992. Hard and Easy Distributions of SAT Problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 459-465.

[12] Morris, P. 1993. The Breakout Method For Escaping From Local Minima. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 40- 45.

[13] Papadimitriou, C. and Yannakakis, M. 1991. Optimization, Approximation, and Complexity Classes. *Journal of Computer and System Sciences* 43: 425-440.

[14] Russell, S. and Wefald, E. 1991. Principles of metareasoning. *Artificial Intelligence* 49: 361-395.

[15] Sandholm, T. 1994. A New Order Parameter for 3SAT. *Twelfth National Conference on Artificial Intelligence Workshop on Experimental Evaluation of Reasoning and Search Methods*, Seattle, Washington.

[16] Sandholm, T. and Lesser, V. 1994. Utility-Based Termination of Anytime Algorithms. In *Proceedings of the European Conference on Artificial Intelligence Workshop on Decision Theory for DAI Applications*, Amsterdam, The Netherlands.

[17] von Winterfeldt, D. and Edwards, W. 1986. Decision analysis and behavioral research. Cambridge University Press.

[18] Williams, C. and Hogg, T. 1993. Extending Deep Structure. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 152-157.

[19] Zilberstein, S. 1993. Operational rationality through compilation of anytime algorithms. Ph.D. dissertation, Department of Computer Science, University of California, Berkeley.