A Model for Projection and Action

Keiji Kanazawa and Thomas Dean*
Department of Computer Science
Brown University, Box 1910
Providence, RI 02912
tld@cs.brown.edu

Abstract

In designing autonomous agents that deal competently with issues involving time and space, there is a tradeoff to be made between guaranteed response-time reactions on the one hand, and flexibility and expressiveness on the other. We propose a model of action with probabilistic reasoning and decision analytic evaluation for use in a layered control architecture. Our model is well suited to tasks that require reasoning about the interaction of behaviors and events in a fixed temporal horizon. Decisions are continuously reevaluated, so that there is no problem with plans becoming obsolete as new information becomes available. In this paper, we are particularly interested in the tradeoffs required to guarantee a fixed reponse time in reasoning about nondeterministic cause-and-eflect relationships. By exploiting approximate decision making processes, we are able to trade accuracy in our predictions for speed in decision making in order to improve expected performance in dynamic situations.

1 Introduction

The world demands behavior that is immediate, and yet guided by the anticipated consequences of observed events. The field is grudgingly coming to accept that systems embedded in the world must be engineered to respond to external events in a timely manner. The consequences of this observation for traditional methods of representation and reasoning in AI are profound. In the *situated automata* approach [Rosenschein, 1987], traditional run-time symbolic reasoning is bypassed in favor of highly specific condition-action relations that have been compiled into circuit-like systems [Rosenschein and Kaelbling, 1986]. However, in domains with complex causal dependencies, particularly nondeterministic ones, the space of possible conditions is large. It may not be feasible to evaluate this space completely at run-time, or to anticipate and store all contingencies at compile-time. In order to generate a run-time system with a guaranteed response time, it may be necessary to make tradeoffs regarding the optimality of the decisions that an agent makes.

*This work was supported in part by the National Science Foundation under grant IRI-8612644 and by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-88-C-0132.

In this paper, we explore a model of decision theoretic control that allows a designer to make such tradeoffs in designing control systems that reason about possible future courses of action at run time. The basic model of decision making is quite simple. At fixed intervals of time, the system makes inferences about all possible plans of actions that it knows about and selects the behavior that maximizes expected utility, given its current knowledge and the time it is given to compute. By fixing the set of possible plans and employing approximate algorithms for computing the expected utility of plans, we are able to trade accuracy in the expected utility computations for speed in responding to the environment and thereby improve the overall system performance. Our methods depend upon being able to perform experiments at design time in order to determine the optimal tradeoff.

The model that we adopt is an extension of our previous work in *probabilistic temporal reasoning* [Dean and Kanazawa, 1988a, Dean and Kanazawa, 1988b]. It is a model of reasoning well suited to tasks such as detecting simple interactions among behaviors and coping with uncertain events. With the adoption of a decision analytic criterion for selecting among behaviors, the control system is able to perform run-time decision making in accord with the tenets of decision theory, and the designer is able to able to make compile-time tradeoffs that take into account realistic computational capabilities and improve expected performance. By selecting the behavior that maximizes utility given its most current state of knowledge, a system based on our model should be able to respond to changing conditions in the world in a timely manner.

2 Our Model of Projection and Action

In our model, an agent has a fixed representation of knowledge about the world in the form of an *influence diagram* [Howard and Matheson, 1984]. An influence diagram is a compact graphical representation of a probabilistic causal theory including the effects of deterministic decisions and preferences over possible world states (see Figure 1). In the following, we follow Shachter's

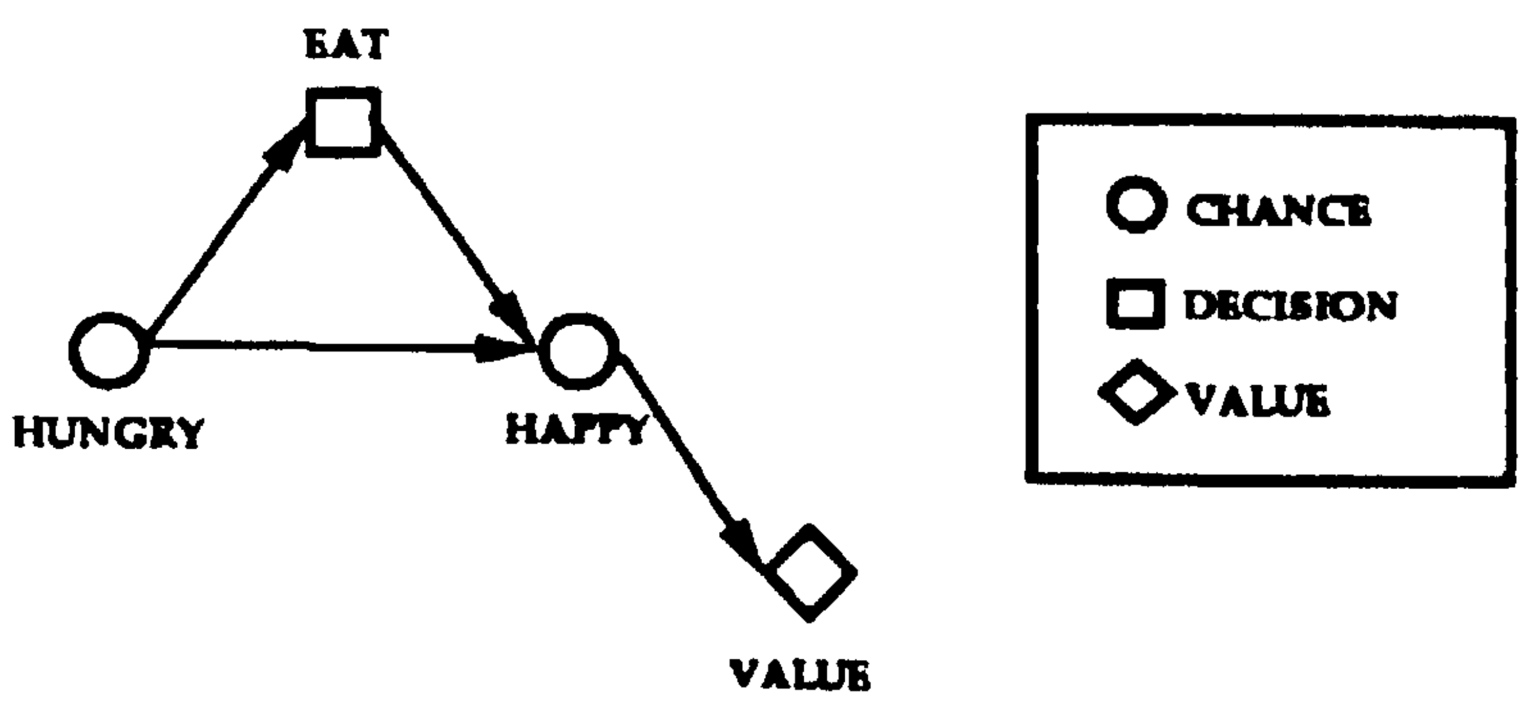

Figure 1: An influence diagram

presentation of influence diagrams in [Shachter, 1986].

An influence diagram is a directed graph $G = (N, A)$, consisting of a set of nodes N, and a set of arcs A. Nodes represent propositions, decisions, and the value of world states; the arcs define the causal and informational dependency between nodes, and the function defining the value of world states.

The arcs define a successor relation between nodes. The set of *direct successors* succ(i) of a node $i \in N$ is defined as $\text{succ}(i) = \{j \in N : (i,j) \in A\}$, while the *indirect successors* of i is the set of nodes along directed paths emanating from node i. Similarly, we may define the *direct predecessors* pred(i) as pred(i) $= \{j \in N : (j,i) \in A\}$, and the *indirect predecessors* as the set of nodes along directed paths into i.

The nodes of an influence diagram consist of a set of *chance nodes* C, a set of *decision nodes D*, and a set of *value nodes V*. Chance nodes are discrete valued variables that encode states of knowledge about the world[1]. Let Ω_i be the set of discrete values of a node $C_i \in C$. There is a probability distribution $P(C_i = \omega, \omega \in \Omega_i)$ for each chance node. If the chance node has no predecessors, then this is its marginal probability distribution; otherwise, it is a conditional probability distribution dependent on the states of its direct predecessors. Decision nodes model the choices of an agent. They are discrete valued; each state of a decision node corresponds to a deterministic choice by an agent to perform a certain action. Value nodes are continuous valued and represent the objective(s) to be maximized in expectation. Value nodes only have arcs going into them; associated with each value node is a *value function* mapping from the states of its direct predecessors to the real line.

Our model of reasoning utilizes a special case of influence diagrams, which we refer to as *causal models.* Causal models explicitly take time into account, in order to allow us to make predictions about the future which we refer to as *projections.* Let $\mathcal{P}$ be a set of propositions of interest in our domain. Define the *time partition T* to be a finite set of *time points,* discrete points in time, that we are interested in representing. The set of chance nodes C in the complete causal model is constructed from $\mathcal{P} \times T$ so that $C = \cup_{t \in T} C_t$, where each C_t is a set of variables, one for each proposition in $\mathcal{P}$ corresponding to that proposition being true at time t.

[1] Influence diagrams with continuous chance variables have been studied [Shachter and Kenley, 1988], but they are generally more complex and will not be considered here.

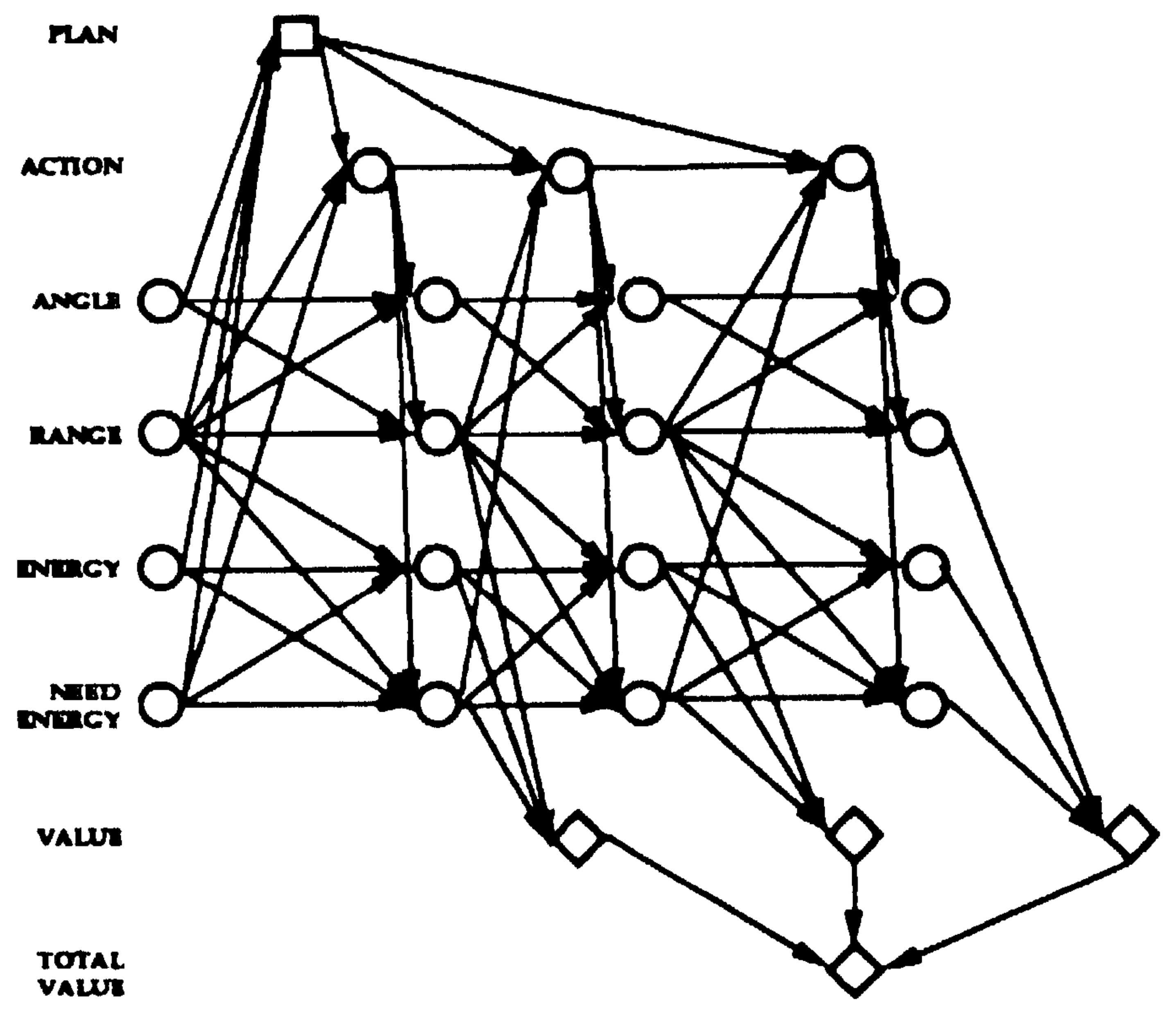

Figure 2: A simple causal model for a situated agent

Figure 2 depicts part of a causal model for a simple autonomous agent. This agent is designed to model a mobile robot that moves about in a simple environment, and tries to stay in operation by replenishing its energy at locations known to have energy sources. The part of the causal model in the figure encodes how certain actions, such as moving, change the robot's position with respect to locations, and models changing expectations about the energy remaining at a given location.

Causal models represent time *indexically;* all time points in the time partition are in reference to the present time, and there is no quantification involving time. A causal model may have a *fixed time model* in which case there is a fixed distance between each point in time, or it may have a *telescopic time model* in which case the interval between consecutive time points is small near the present time, but becomes gradually larger between consecutive time points in the future. Within a causal model, there are two distinguished time points, one representing past information and one current sensory inputs; all other time points correspond to those in the future. It is often important to keep information from the past for various purposes, for example, for incorporating sensor data from the past.

Within a causal model, a special class of influences describe *persistence* [McDermott, 1982] of propositions. We may, for example, have knowledge such as, *"The energy at a given source is likely to diminish by 10% every hour".* Let $P(energy, t)$ be the probability that a energy exists at time t; one way to encode the diminishing probability of energy remaining is by the conditional probability $P(energy, t+60 | energy, t) = 0.9$.

Within the scope of this work, causal models are distinguished by a Markov property and a *time separa-*

ble value function. The Markov property means that a proposition is only affected by propositions from the previous time point. Thus,

$$C_t \supset \bigcup_{c \in C_{t+1}} \mathrm{pred}(c).$$

By a time separable value function, we mean the following. Let us define a set of value nodes $\{V_t : t \in T\}$. Assume that for each time point t, there exists a function u_t such that $V_t = u_t(C_t^v), C_t^v \subseteq C_t$. We say that our value function is time separable if the total value V is given by

$$V = \sum_t V_t.$$

In that case, we say that V_t is the *objective value at time t.* These two properties essentially define a *Markov decision process*[2] [Howard, 1960]. As far as decisions in causal models are concerned, within the scope of this work, causal models contain only one decision: the plan selection decision.

A plan in our model is just a sequence of *behaviors* where a behavior corresponds to a mode of operation for a subsystem in a layered control architecture [Brooks, 1985, Kaelbling, 1987]. Each behavior in a plan has a condition attached to it called its *task completion criterion* that specifies when to terminate the behavior. For example, consider the plan [wander until near-energy-source] // [eat until 'hungry] // [wander forever], where // is a sequencing operator. The first step in the plan says to wander until the agent is near an energy source. Within the causal model, we assume that there is a proposition that represents the current action being undertaken at a point in time. In concert with that and the task completion criterion, we are able to determine at what point in the causal model a particular subtask might end and another go into effect. Note that, since influence diagrams are static representations, plan generation is carried out at design time.

Given a causal model, the agent initiates the behavior specified in the plan with the highest expected utility. There is, however, no commitment to a specific plan; only the first step in a particular plan. As soon as an agent has initiated the first behavior in a plan, it is recomputing the plan with the highest expected utility based on whatever new information is available. Suppose there is a plan [P] // [Q]. The agent may choose [P] // [Q] as the best plan, but once it has actually completed [P], it will not actually begin to do [Q], unless there is a separate plan [Q]. Because of this, every plan suffix must itself be represented as a plan.

Let us now define the *causal model decision problem.* The *input nodes* for a causal model decision problem consiste of the set of chance nodes representing past information, and those representing current sensory information. The *input* to the run-time system is a probability distribution on the input nodes. Recall that there is a single decision node representing the plan selection decision in causal models. The solution to a causal model decision problem is the plan with the maximum expected utility.

Thus, at any given instant, an agent has a set of possible plans, and a prior distribution on the nodes corresponding to the current sensor input. An agent with unlimited computational capabilities would compute the expected utility of each plan, choose the plan with maximum utility, and then initiate the behavior corresponding to the first step in the chosen plan, and it would repeat this as often as necessary to keep pace with changes in the information returned by its sensors. Assuming that an agent has limited computational capabilities, it takes some time to compute the plan that maximizes expected utility; this time determines the *response time* of the agent: how fast the agent responds to changes in its input.

An influence diagram consisting only of chance nodes is known as a *belief net.* Roughly speaking, since the expected value for an influence diagram is uniquely determined by the conditional probability distribution of its chance nodes given deterministic choices and prior distributions in the root nodes of the graph, if there are efficient methods for computing a distribution in a belief net, then it will be efficient to compute the expected value for an influence diagram. Unfortunately, it has been shown [Cooper, 1988], that that computing a probability distribution for a general belief network is an NP-Hard problem. Even the most efficient known algorithm for computing an exact distribution in a belief net [Lauritzen and Spiegelhalter, 1988] has a time complexity exponential in the size of the largest clique in the belief net graph [3]. Therefore, in general, it may be necessary to make tradeoffs in order to obtain guaranteed response-time behavior.

3 Tradeoffs in Generating Efficient Run-Time Systems

Our goal is to construct embedded systems with a procedure, or a set of procedures, that solves the causal model decision problem in bounded time. We speak of the process of constructing such a procedure as *compilation]* the product, of compilation is referred to as the *run-time system.* In this section, we explore various tradeoffs that we may need to make in order to construct an efficient run-time system.

To make the tradeoffs involved in compilation precise, we need to consider the overall utility of a run-time system including the practical utility associated with the time that is spent in computing an answer [Dean and Boddy, 1988, Horvitz, 1988). A run-time system that computes accurate answers, but takes a long time to do so is not likely to be as useful as a system that computes approximate, but close to accurate answers very

[2]It might be appropriate the call causal models Markov influence diagrams; however, a Markov influence diagram would be a proper superset of what we have termed causal models because causal models are defined to have a time separable value function.

[3]There is a known efficient polynomial time algorithm for computing the probability distribution for a belief net if the underlying *undirected* graph of the net has no cycles [Pearl, 1988].

fast. Our goal in compilation is to maximize the overall utility.

As an illustration of a simple compilation method, consider the following. Assume that we have a number of algorithms available to us for computing an answer to the problem of what to do next. Each of these algorithms has a parameter that can be set to enable the algorithm to return an answer in a fixed amount of time. We fix the causal model and determine a set of *cycle intervals* (intervals of time that we will allow for computation) and a set of possible-world simulations that we will use for evaluation. For each algorithm and cycle interval, we run all the possible-world simulations and obtain a cumulative score. We then select the algorithm and cycle interval that has the best score to use in the run-time system. Simple as this may seem, compile-time decisions based on simulations are at the heart of most engineering approaches to control. We,now consider how one might actually generate an appropriate set of algorithms for performing this sort of compilation.

There are three basic methods that are available to us in order to obtain better response time. These are off-line computation, model reduction, and approximate run-time computation. The tradeoffs themselves come in two basic flavors: space/time and accuracy/time. In the best cases, we will be able to effect a problem reformulation that allows us to maximize our overall utility without sacrificing accuracy. This will most likely be achieved at the cost of a combination of off-line computation and run-time storage. In other cases, we may need to sacrifice the accuracy of the run-time system to produce acceptable overall utility. In these cases, the run-time system will be an *approximation scheme.* Where the approximation scheme involves a run-time computation algorithm, we may speak of the algorithm as an *approximation algorithm.*

As noted above, in the ideal cases, it will be possible to construct either a circuit, a table, or a reduced model that enables us to compute exact answers with no loss in accuracy. In general, since space may grow exponentially, there are limits to the applicability of such methods. However, where the cost of memory or circuitry is cheap, or the cost of slow performance is great, these methods are likely to produce the most significant gains in performance. Later in this section, we examine a method of causal model reduction which produces substantial performance improvements.

A special class of approximation algorithms receiving attention are *anytime algorithms* [Dean and Boddy, 1988, Boddy and Dean, 1989]: algorithms that iteratively improve the quality of their answers relative to a given query. Such algorithms are of particular value in *deliberation scheduling,* where an attempt is made at run-time to maximize overall utility of a combination of deliberation and action. Since more computation may result in a better answer, there is a tradeoff to be made between acting and spending more time computing.

Our approach to compilation determines at compile-time the fixed time for deliberation that results in the maximal overall expected utility. Within that context, anytime algorithms are just as applicable for us as for systems with more complex run-time deliberation scheduling. Applicable anytime algorithms for causal models include Monte Carlo simulation algorithms [Henrion, 1988a, Pearl, 1988], and bounding algorithms such as bounded cutset conditioning [Horvitz *et* a/., 1989].

So far, we have not said much about the plan selection decision itself. As we noted before, the causal model decision problem involves selecting the plan with the highest expected utility. There is substantial opportunity to exploit context and prior expectation of the value of adopting various plans, and the properties of anytime algorithms in this process. We may determine an order of plan evaluation with high expected utility depending on context and prior expectation of the value of various plans in that context. The attraction of anytime approximation algorithms, and particularly anytime algorithms with guaranteed bounds on their answers is that they can be used to quickly cycle through the space of decisions and identify those decisions that appear to be most worthwhile pursuing. If the lower bound of the expected value of one decision is more than the upper bound of another, then we can drop consideration of the second alternative and focus on the first decision and others.

It is also possible to combine methods such as table generation, with approximation algorithms to obtain even better speedups. For example we may choose to tabulate the optimal decisions for the most time-critical situations, or the most frequently ocurring situations in our domain.

Finally, in cases where a combination of model reduction and approximation algorithms are either infeasible or do not produce the desired results, we may need to sacrifice accuracy in the model, producing an *approximate model.* This may be done through a sensitivity analysis on the causal model to determine what input nodes and which decisions the expected value is most sensitive to. As a result, we may reduce the number of nodes, node states, or influences in the causal model. We may also, for example, compile a table that contains the most important and most likely decisions that the run-time system will need to make. Such methods are related to the use of defaults in nonmonotonic reasoning schemes. Unfortunately, we do not have the space to explore issues in the use of approximate models fully in this paper.

We now focus attention on a particular method, *model reduction through absorption,* that offers large reductions in the time-complexity of run-time systems for moderate-sized causal models. A particularly attractive topology for causal models for computational purposes is an influence diagram that maps directly from the inputs to a single value node. In the causal model decision problem, we are essentially uninterested in the actual states of chance nodes at future time points, except inasmuch as they contribute to the expected utility. Therefore, there is nothing conceptually to bar us from reducing our causal model to this form (see Figure 3).

There is a well known method [Shachter, 1986] for eliminating a chance node in an influence diagram when it is neither an input node, nor a node for which we are explicitly computing a conditional probability distribu-

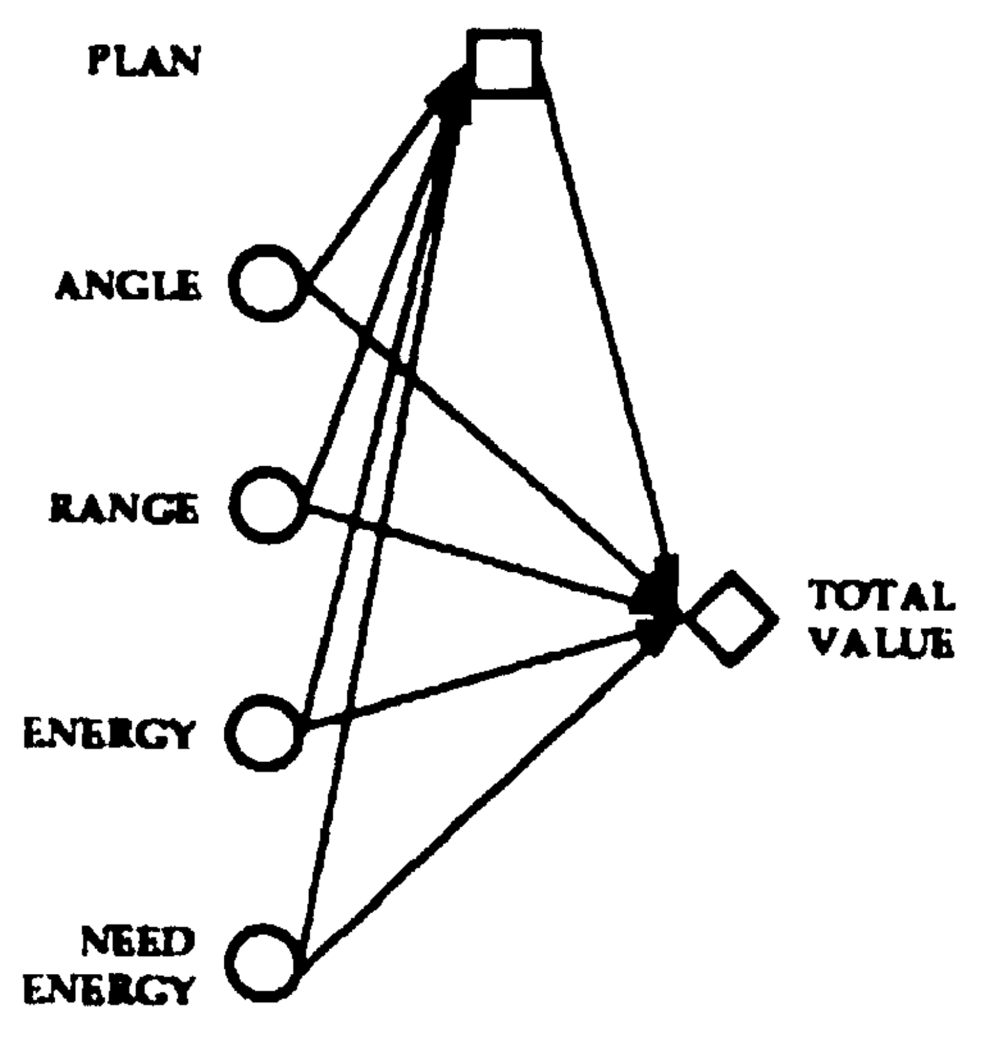

Figure 3: A reduced causal model for a situated agent

tion. This is the process of *absorption;* when a chance node is absorbed, its successors inherit the chance node's predecessors, and a new influence is computed by conditional expectation on the states of the absorbed chance node.

Such reduced influence diagrams have very attractive properties. Although it may still take exponential time to compute exactly with such networks, it will be of a drastically reduced order from the original network, rendering it more tractable to compute. Time complexity in the reduced causal model may be further reduced if the inputs nodes are independent of one another; then the reduced model becomes tree-like with no influences between input nodes. The time complexity of the reduction is itself exponential in the size of the set of propositions. We ran a series of experiments on causal models which in their original form experienced serious combinatorial explosion in exact computation. In one set of experiments, increasing the size of the causal model by one time point caused the exact computation time to increase from 15 minutes to 2 hours, and by adding another time point, from 2 hours to over 24 hours. The time for exact computation decreased by a couple of orders of magnitude when reduction was applied to the causal model. When we increased the number of time points in the causal model, the post-reduction exact computation time increased only roughly linearly. The run-time of the reduction process itself was of the same order of magnitude as the post-reduction exact computation time.

Reduced causal models of this form are attractive for other reasons. At compile-time, reduction of the model in this form makes the causal model easier to analyze and apply other compilation methods including further reduction of the model through sensitivity analysis. At run-time, the reduction of the model lead to performance improvements not only for exact algorithms, but for approximation algorithms as well.

In experiments that we have conducted with Monte Carlo simulation algorithms, the probability for input nodes typically converges very quickly, but the time it takes to converge for leaf nodes appears to increase substantially as depth in the circuit grows. Reducing the depth of the network offered substantial speedup in the convergence of the value node. For bounding algorithms, the topology of reduced diagrams makes it easy to generate good heuristic methods of ordering the plan evaluation. We are currently performing experiments in this area.

Finally, the reduced diagram is in a form that can be separated into multiple copies without a increasing the size of the value function. Because the value node depends on the value of all input nodes and the decision node, we may separate out the value function for each decision. Except for the overhead of the size of nodes, there is no net increase in space. With different copies for each decision, it is possible to interrupt expected value computation for a decision without overhead in swapping intermediate results. Therefore, depending on how well the expected value is converging for each decision alternative, we may focus our computational resources on those decisions that appear to clearly dominate others. Naturally, it would be possible to compute the expected value for each decision in parallel as well.

The space tradeoff involved in this diagram reduction is as follows. The value function for any value node increases (at least doubles) for every increase of one in the size of the set of predecessors to the value node. Since a value node inherits all of its predecessors, the value function may grow fairly large as we "roll back" the value nodes toward the input nodes. A value node ultimately inherits all of its predecessors in the set of input nodes. At the very last stage, we have a situation where all the value nodes are direct successors of the input nodes [4]. We can replace the $||T||$ value function tables by one, since the total objective value is just the sum of the objective values at each time step. Because of the regular structure of causal models, we are guaranteed that this table will be no larger than the largest of the ||T|| tables. If the largest table is less than the product of the size of the original value function tables (note that they are all the same) and the number of times points, then we have a net reduction in space. Thus, the space penalty incurred by reduction by absorption appears relatively benign for causal models.

4 Discussion

There are a number of problems with the approach as it is outlined here. The current model does not handle continuous variables at all. In particular, it is not possible to do any sophisticated spatial reasoning within our framework. Our approach does not allow an agent to take advantage of situations in which it can make reasonably accurate long term predictions: situations in which the cost of planning might be amortized over some length of time. In the classical approach to planning, an agent computes a plan once, and commits to the plan to acnieve a goal. This can be advantageous if the world is relatively static, or if the agent has an effective method for predicting future states; in such cases, the work done in generating a plan need only be done once. In a more dynamic environment, committing to a plan can be problematic; a situation can change quickly rendering a plan obsolete that was considered optimal under some previ-

[4] Note that any input node that is not a predecessor of a value node can be eliminated at this stage.

ous state of knowledge. In the approach taken in this paper, an agent recomputes the best plan at regular intervals. More complicated strategies may be necessary to achieve a desired level of performance.

Despite the above shortcomings, the approach outlined in this paper directly addresses a number of important tradeoffs concerning the value of prediction in dynamic situations that have motivated earlier work but have never been explicitly spelled out. It is our contention that the only way to make sense of these tradeoffs is within a decision theoretic framework. We see our approach as providing a connection between the symbolic processing approach of AI and those disciplines that emphasize real-time control of processes.

The main contribution of this paper is to provide a model for control that incorporates run-time reasoning about possible futures to support plan selection in dynamic environments. By exploiting approximate decision making processes, we are able to trade accuracy in our predictions for speed in decision making in order to improve expected performance. With the adoption of a decision analytic criterion for selecting among plans, the control system is able to perform run-time decision making in accord with the tenets of decision theory, and the designer is able to able to make compile-time tradeoffs that take into account realistic computational capabilities. By continually attempting to determine a plan that maximizes utility, a system based on our model should be able to respond to changing conditions in a timely manner while at the same time taking into account future states and assessing the value of extended plans of action.

References

Boddy and Dean, 1989] Mark Boddy and Thomas Dean. Solving time dependent planning problems. In *Proceedings IJCAI-89,* 1989. (In this volume).

[Brooks, 1985] Rodney A. Brooks. A robust layered control system for a mobile robot. A. I. Memo 864, MIT Artificial Intelligence Laboratory, Cambridge, Massachusetts, 1985.

[Cooper, 1988] Gregory F. Cooper. The computational complexity of probabilistic inference using belief networks. Memo KSL-87-27, Knowledge Systems Lab, Stanford University, 1988.

[Dean and Boddy, 1988]
Thomas Dean and Mark Boddy. An analysis of time dependent planning. In *Proceedings AAAI-88,* 1988.

[Dean and Kanazawa, 1988a] Thomas Dean and Keiji Kanazawa. Probabilistic causal reasoning. In *Proceedings of the Canadian Society for Computational Studies of Intelligence.* CSCSI, 1988.

[Dean and Kanazawa, 1988b] Thomas Dean and Keiji Kanazawa. Probabilistic temporal reasoning. In *Proceedings AAAI-88.* AAAI, 1988.

Henrion. 1988a] Max Henrion. Propagating uncertainty by logic sampling in bayes' networks. In John F. Lemmer and Laveen F. Kanal, editors, *Uncertainty in Artificial Intelligence 2,* pages 149-163. North-Holland, 1988.

[Henrion, 1988b] Max Henrion. Towards efficient probabilistic diagnosis in multiply connected belief networks. In *Proceedings of the Conference on Influence Diagrams,* Berkeley, CA, 1988.

[Horvitz *et al,* 1989] Eric J. Horvitz, Gregory F. Cooper, and David E. Heckerman. Reflection and action under scarce resources: Theoretical principles and empirical study. In *Proceedings IJCAI-89,* 1989. (In this volume).

[Horvitz, 1988] Eric J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings AAAI-88,* pages 111-116, 1988.

Howard and Matheson, 1984] Ron A. Howard and James E. Matheson. Influence diagrams. In Ron A. Howard and James E. Matheson, editors, *The Principles and Applications of Decision Analysis.* Strategic Decisions Group, Menlo Park, CA 94025, 1984.

[Howard, I960] Ron A. Howard. *Dynamic Programming and Markov Decision Processes.* MIT Press, 1960.

[Kaelbling, 1987] Leslie Pack Kaelbling. An architecture for intelligence reactive systems. In Michael P. Georgeff and Amy L. Lansky, editors, *Reasoning About Actions and Plans,* pages 395-410. Morgan Kaufmann, 1987.

Lauritzen and Spiegelhalter, 1988] Stephen L. Lauritzen and David J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society Series B,* 50(2): 157-194, 1988.

McDermott, 1982] Drew V. McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science,* 6:101-155, 1982.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufman, 1988.

[Rosenschein and Kaelbling, 1986] Stanley J. Rosenschein and Leslie Pack Kaelbling. The synthesis of digital machines with provable epistemic properties. In *Proceedings Conference on Theoretical Aspects of Reasoning About Knowledge,* pages 83-98, Asilomar, California, 1986.

[Rosenschein, 1987] Stanley J. Rosenschein. Formal theories of knowledge in ai and robotics. Report No. CSLI-87-84, Center for Study of Language and Information, Stanford, California, 1987.

[Shachter and Kenley, 1988
Ross D. Shachter and C. Robert Kenley. Gaussian influence diagrams. *Management Science,* 1988. (To appear).

[Shachter, 1986] Ross D. Shachter. Evaluating influence diagrams. *Operations Research,* 34(6):871~882, November/December 1986.