Prediction, Observation, and Estimation in Planning and Control

Thomas Dean* Keiji Kanazawa John Shewchuk
Department of Computer Science
Brown University, Box 1910, Providence, RI 02912

Abstract

We begin by looking at the central role of reasoning about change in planning and control. We then review some of the basic insights of control theory regarding the connection between observability and controllability, and consider how these insights should inform research in planning. We propose a decision-theoretic approach to planning that subscribes to a basic cycle of prediction, observation, estimation, and action selection. We examine some of the practical issues relating to this approach, and consider alternatives in the form of off-line compilation of decision models and learning systems to circumvent the need for accurate predictive models. We illustrate our points by considering problems in mobile robotics that require integrating obstacle avoidance, position estimation, and path planning.

Introduction

Predictive models are used in the design of planning and control systems to describe the behavior of dynamical systems; they also are often used as components in planning and control systems to allow a system to predict its own behavior and that of the environment in which it is embedded. In this paper, we consider some of the issues involved in making effective use of predictive models in planning and control systems.

There are several formal approaches to modeling dynamical systems. The differential and integral calculus, temporal logic, finite state automata, and stochastic processes have all been used for applications that require reasoning about time and change. It is still mostly an art to develop a useful predictive model for any particular dynamical system, where by useful I mean that the model provides information to assist in generating control actions without which the system as a whole would perform less well. The model itself is just a formal object; for effective control, not only do we need a useful model, but we need a realization of such a model as a computational process that can provide the necessary information in a timely manner.

In the following, we explore two main issues. First, prediction by itself is generally not enough. There is usually so much uncertainty in our predictions that it is necessary to make frequent observations to align our predictions with reality. Among researchers in systems control, the use of feedback is common practice. However, many researchers in automated planning are just beginning to make use of feedback, having spent a great deal of their effort to date on domains that are completely predictable. Second, computational costs cannot be ignored in formulating a suitable model. Many real-world dynamical systems can be quite complex to model; this representational complexity results in computational complexity in the algorithms used for realizing these models for practical applications. A designer has to weigh the informational benefits of a model against the costs due to delays in computing the predictions of the model.

Accounting for Uncertainty

All predictive models have some limitations. Even if the model accurately mirrors the real world, the measurements made to set up the initial conditions for the model are often subject to errors. Such errors can result in weak or false predictions depending on whether the model accounts for them. Uncertainty in the measurements used to provide initial conditions is magnified as the predictions extend further into the future.

In many applications, sensing is reasonably inexpensive. If we account for measurement errors, making frequent measurements can substantially improve the accuracy of prediction. In some cases, it is sufficient for good performance to obtain information about the current state; either future states do not matter, or knowledge of the current state is sufficient for predicting relevant future states. In other cases, it is necessary to combine information from past states with that

*This work was supported in part by a National Science Foundation Presidential Young Investigator Award IRI-8957601 with matching funds from IBM, 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.

from the current state in order to achieve good performance. This idea of combining past and present information suggests a cycle of activity typical of many control strategies:

- Predict — generate expectations regarding future states using the current model.
- Observe — gather information in an effort to confirm/disconfirm expectations and identify unexpected events.
- Estimate — combine observations and expectations to update the current model providing estimates for current and future states.
- Control — act on the basis of your estimates.

In this paper we view planning in terms of enumerating a set of possible courses of action, evaluating the consequences of those courses of action, and selecting a course of action whose consequences maximize the expectation of a given performance (or *value*) function. From this decision theoretic perspective, there is no difference between an observation (i.e., an action that involves sensing) and any other action; a decision maker simply tries to choose actions that maximize expected value. Given this unification, the above cycle reduces to one alternating between predicting future states on the basis of past observations and selecting actions so as to maximize expected value.

Many existing planning systems just assume that they are provided with an accurate model and initial conditions [Wilkins, 1984, Dean *et al.*, 1988]. Most of the models are based on first-order predicate logic with some sort of temporal machinery. These models are not particularly well suited to reasoning about uncertainty in either sensing or prediction. While there have been some efforts to explore the use of decision and probability theory in planning [Langlotz *et al.*, 1987, Feldman and Sproull, 1977], it is only recently that researchers have begun to explore methods for reasoning about time that account for uncertainty in ways that make them suitable for complex planning applications [Cooper *et al.*, 1988, Dean and Kanazawa, 1988, Hanks, 1988, Weber, 1989].

In the next section, we consider an approach to reasoning about time and action that is well suited to dealing with uncertainty in sensing and prediction, and that allows us to capture the cycle activity outlined above.

Reasoning About Time and Action

A *Bayesian network* [Pearl, 1988] is a directed graph $G = (V, E)$. The vertices in V correspond to random variables and are often referred to as *chance nodes*; the edges in E define the causal and informational dependencies between the random variables. In the model described in this paper, chance nodes are discrete valued variables that encode states of knowledge about the world. Let Ω_C be the set of discrete values of a chance node C. There is a probability distribution

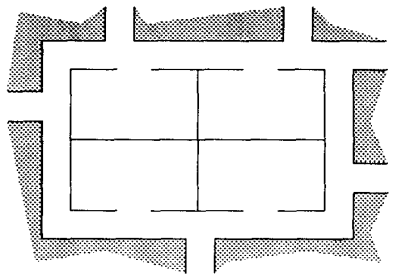

Figure 1: Layout of an office environment

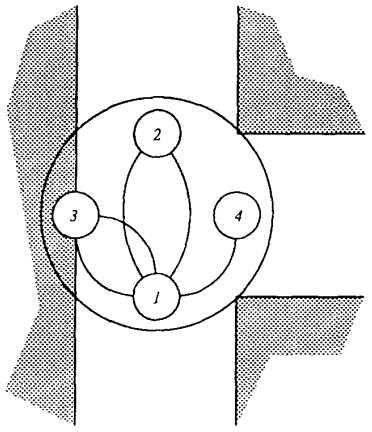

Figure 2: Classifying locally distinctive places

$\Pr(C = \omega, \omega \in \Omega_C)$ for each 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 the immediate predecessors of C in G. In the following, we consider a special case of Bayesian networks for sequential decision making, but we begin with a simple example.

Imagine a mobile robot assigned the task of delivering packages in an office environment. To facilitate navigation in such an environment, the robot distinguishes between corridors and the various types of junctions where corridors meet (see Figure 1). Distinguishing between, say, a T junction and an L junction can involve several measurements using the robot's ultrasonic sensors. Each measurement may require the robot to move to a different location. For instance, Figure 2 shows a sequence of steps taken in classifying a particular junction. To illustrate the application of Bayesian networks for control, we construct a probabilistic model to assist the robot in deciding where to move and what to sense next.

Let H be a random variable that takes on values from a set of junction types, and $X_{f,w}$ be a random variable used to represent whether or not the feature f

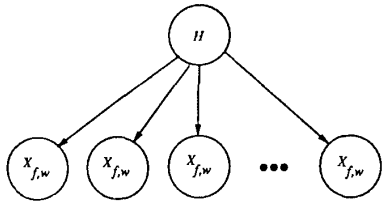

Figure 3: Bayesian network model for classification

is present at the location w. Each measurement consists of moving to a particular location w and scanning for a feature f, thus providing evidence for or against $X_{f,w}$. Figure 3 shows a Bayesian network representing how the evidence obtained from measurements influences our beliefs about the type of junction. To quantify the model, we need $\Pr(X_{f,w}|H)$ which is obtained directly from the geometry and $\Pr(H)$ which is our a priori knowledge about the distribution of junction types. The model shown in Figure 3 assists in deciding what action to take next; it also allows us to update the model as measurements are taken. Having already performed n measurements corresponding to the sequence of feature/location pairs $\langle f_1, w_1\rangle, \ldots, \langle f_n, w_n\rangle$,

$$\Pr(H|X_{f_1,w_1}=\sigma_1, \ldots, X_{f_n,w_n}=\sigma_n)$$

represents the posterior distribution on H given the data $\sigma_1, \ldots, \sigma_n$. In [Dean *et al.*, 1990], we provide a decision model based on the simple Bayesian network shown in Figure 3. The decision model selects the measurement that is most discriminating (i.e., maximizes the separation in H).

In the previous model, we were only interested in selecting a single action using a selection criteria that did not require reasoning about future states of the world. In order to reason about sequences of actions, we have to reason about future states. One way to do so is simply to replicate the variables corresponding to the current state of the world to capture future states. This is essentially what we do in our *temporal Bayesian network* formulation [Dean and Kanazawa, 1989].

Given a set of discrete variables, $\mathcal{X}$, and a finite ordered set of time points, $\mathcal{T}$, we construct a set of chance nodes, $\mathcal{C} = \mathcal{X} \times \mathcal{T}$, where each element of $\mathcal{C}$ corresponds to the value of some particular $x \in \mathcal{X}$ at some $t \in \mathcal{T}$. To illustrate, consider the following problem.

Consider a mobile robot required to track targets in a cluttered environment. The robot is provided with sonar and rudimentary vision. The robot's task is to detect and track moving objects, reporting their location in the coordinate system of a global map. Initially, the robot systematically explores its surroundings scanning for moving objects. Once it detects such an object, it is required to track the object and report on its location. The robot is supplied with a floor plan of the

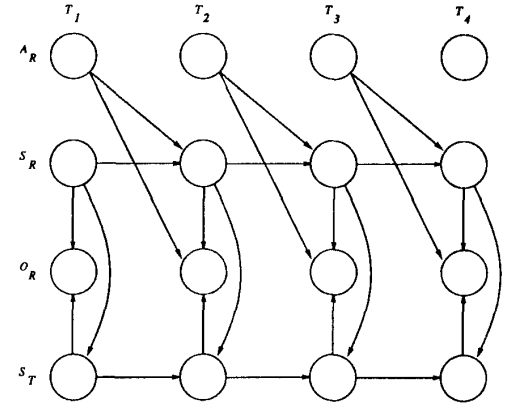

Figure 4: Probabilistic model for the tracking problem

office showing the position of permanent walls and major pieces of furniture such as desks and tables.

We assume that there is error in the robot's movement requiring it to continually estimate its position with respect to the floor plan so as not to become lost. Position estimation (*localization*) is performed by having the robot track *beacons* corresponding to walls and corners and then using these beacons to reduce error in its position estimate. Localization and tracking are frequently at odds with one another. A particular localization strategy may reduce position errors while making tracking impossible, or improve tracking while losing registration with the global map.

Let S_R and S_T be variables ranging over the possible locations of the robot and the target respectively. Similarly, let A_R be a variable ranging over the actions available to the robot. At any given point in time, the robot can make observations regarding its position with respect to the global map and the target's position with respect to the robot; O_R is a variable ranging over sets of such observations.

Figure 4 shows a temporal belief network for $\mathcal{X} = \{S_R, S_T, A_R, O_R\}$ and $\mathcal{T} = \{T_1, T_2, T_2, T_4\}$. To quantify the model shown in Figure 4, we have to provide distributions for each of the variables in $\mathcal{X} \times \mathcal{T}$. We assume that the model does not depend on time, and, hence, we need only provide one probability distribution for each $x \in \mathcal{X}$. The numbers for the probability distributions can be obtained by experimentation without regard to any particular global map.

Assuming the state-of-the-art in algorithms for evaluating Bayesian networks, we can easily determine the cost of evaluating a model based on such networks. The cost of evaluating the network shown in Figure 4 is dominated by the product of $|\Omega_{S_R}|$ and $|\Omega_{S_T}|$: the sizes of the sample spaces for S_R and S_T. If the representation for the locations of the robot and the target is too fine grained then $|\Omega_{S_R}|$ and $|\Omega_{S_T}|$ can easily get out of

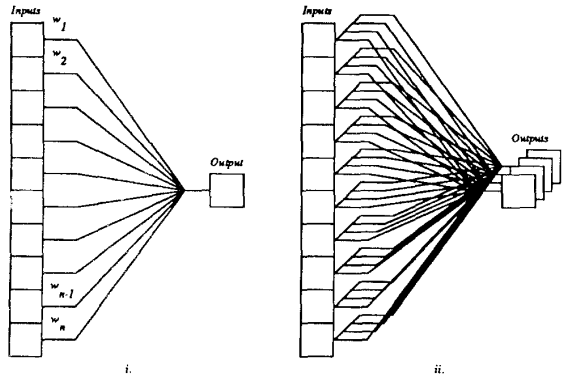

Figure 5: Single and multiple output linear models

use them to look up an entry in the table. The robot would then select the target-position-estimate/robot-action pair with the largest expected value given the expectation stored in the entry. For certain control problems, simple error-minimization networks (see Figure 5.i) can be extended to handle multiple outputs for representing the expectation over $\Omega_{S_T} \times \Omega_{A_R}$ (see Figure 5.ii) suffice [Shewchuk and Viola, 1989].

In most cases, the target's position and the correct action will not be completely characterized by the current observations. However, if we assume a k-Markov property for the process governing the state variables, it is sufficient for a complete characterization to learn a mapping from $(\Omega_{O_R} \times \Omega_{A_R})^k$ to $\Omega_{S_T} \times \Omega_{A_R}$. Although it is unlikely that we could build the resulting table for a realistic problem, there are techniques that can be applied to make the problem more tractable.

If we assume that only some of the entries in the table are utilized, then we can employ hashing techniques to reduce the storage requirements for the table. This appears to be a reasonable assumption given that the geometry of the world does not allow all possible combinations of action and observation sequences, and that, if the robot is trying to accomplish some plan, then robot's actions will further restrict the sequence of actions and observations. This is the assumption made in Albus' [1975] CMAC model, as well as in the sparse matrix techniques that are employed in reducing computational complexity in Bayesian networks. In Albus' model, a large input space is mapped into a smaller memory space using hashing techniques (see Figure 6). In Shewchuk and Dean [1990] (this proceedings), we present a variation on Albus' model that enables a system to quickly adapt as environmental conditions change.

Another technique to reduce storage requirements would be to combine several of the observations, using sensor fusion techniques. Letting S_R represent our current best estimate of the robot's location, we can

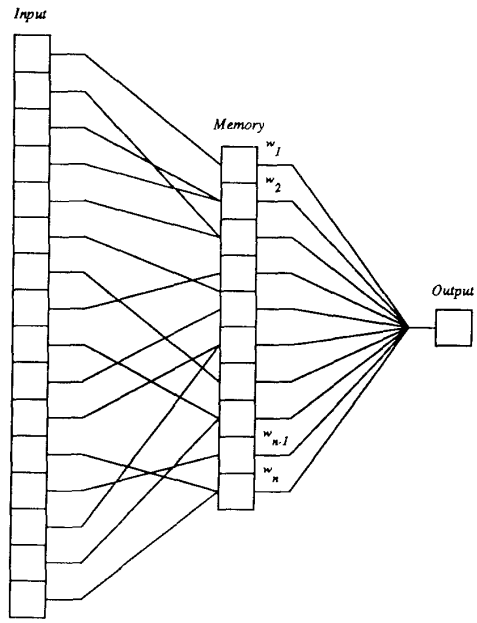

Figure 6: Compressing the input space in Albus' model

reformulate the problem in terms of two mappings. The first mapping handles the sensor fusion calculation: $\Omega_{O_R} \times \Omega_{S_R}$ to Ω_{S_R}. The second mapping handles the target-position-estimate/robot-action calculation: $\Omega_{O_R} \times \Omega_{S_R}$ to $\Omega_{S_T} \times \Omega_{A_R}$. This intermediate mapping scheme enables us to avoid some of the additional memory requirements that occur as a consequence of having to account for the last k observations required by the k-Markov assumption.

There has lately been recent interest in learning techniques for sequential decision making problems [Sutton, 1990, Watkins, 1989, Whitehead and Ballard, 1990]. Many of the proposed approaches rely on techniques closely associated with traditional dynamic programming [Bellman, 1957]. We expect that the learning approaches will be subject to many of the same sources of complexity that plague dynamic programming and Bayesian network formulations. We believe that the Bayesian network approach can be combined with current learning approaches to build systems that begin with a great deal of domain specific knowledge which is refined and extended over time. Such hybrid approaches will be necessary for implementing practical control systems (e.g., Sutton [1990] discusses how a prediction component can dramatically speed learning).

The choice of what to learn at run time and what to compile at design time into fixed decision models will depend on the availability and volatility of domain knowledge. The hard problems have little to do with the particular approach to sequencing and inference,

hand, making evaluation prohibitively expensive. On the other hand, if these representations are too coarse grained, then the precision of the model will be reduced thus lowering performance.

As an alternative to using a coarse grained spatial representation, we might modify Ω_{S_R} and Ω_{S_T} on the fly by restricting the range of S_R and S_T on the basis of evidence not accounted for in the decision model (e.g., odometry and compass information). For instance, if we know that the robot is in one of two locations at time 0 and the robot can move at most a single location during a given time step, then $\langle S_R, 0\rangle$ ranges over the two locations, and, for $i > 0$, $\langle S_R, i\rangle$ need only range over the locations in or adjacent to those in $\langle S_R, i-1\rangle$. Similar restrictions can be obtained for S_T. For models with limited lookahead (i.e., small $|\mathcal{T}|$), these restrictions can result in significant computational savings.

It should be noted that another representation for the model shown in Figure 4 is as a Markov chain [Dean and Kanazawa, 1989]. Given such a representation, we might use a dynamic programming method such as Howard's [1960] *policy iteration* scheme to compute an optimal control rule for the underlying sequential decision problem.[1] The cost of policy iteration is polynomial in the size of the state space for the Markov process, which in this case would be dominated, again, by the product of $|\Omega_{S_R}|$ and $|\Omega_{S_T}|$.

The idea of generating an optimal control rule can be applied to Bayesian networks as well. This involves the off-line compilation of a lookup table directly from the Bayesian network. Heckerman *et al* [1989] outline a method for compiling such a lookup table. Unfortunately, their method works only for a restricted form of network, and does not appear to extend easily to the sort of networks that arise in practice. Other methods for compiling a lookup table directly from a Bayesian network involves simulation of the different input states to the decision model. Again, this method is highly dependent on the product of $|\Omega_{S_R}|$ and $|\Omega_{S_T}|$, and suffers the same restriction as policy iteration.

An alternative method for improving the run-time performance of computation with Bayesian networks involves the application of off-line computation to convert the original probabilistic model to a more tractable form. This is the process of *network reduction* [Kanazawa and Dean, 1989]. For a given probabilistic model, we are only interested in a subset of the nodes at run-time: nodes for which we will have or need information at run-time (e.g., observation nodes and nodes concerning the current location of the target). As far as procedures that utilize the computations from such Bayesian networks are concerned, it matters little if other nodes are removed from the network (e.g., nodes corresponding to the actions of the target).

Fortunately, there are a set of well-known operations that can be performed on a Bayesian network by which nodes can be successively eliminated, without affecting the consistency of the network (i.e., the probability distribution over the remaining nodes in the network) [Shachter, 1986]. In network reduction, a series of such *absorptions* of nodes are performed, producing a smaller Bayesian network. Absorption may be performed until there are only input and output nodes of interest remain, or some acceptable measure in the complexity of the network is reached.

In the approaches to sequential decision being considered here, the primary components include a predictive model (e.g., a temporal Bayesian network or a Markov process), a performance criterion (e.g., a value function), and some sort of generator for candidate sequences of actions. In the following, we consider reducing our reliance on the predictive model by allowing the robot's experience to serve as a guide in formulating an optimal policy.

Learning and Decision Making

There are several reasons to consider the application of learning techniques to problems such as those described in the previous section:

- By learning regularities of the environment, a learning system might be able to reduce the computation required for control.
- If environmental conditions change over time, a learning system might be able to adapt to the changing conditions.
- For performance criteria that are difficult to specify, reinforcement learning techniques can simplify the difficult task of writing control programs.

To explore some of the issues in learning for sequential decision making, we reformulate the Bayesian network approach to the mobile target tracking problem in terms of learning input/output functions.

Assume for the moment that the inputs to the learning algorithm consist of an observation about the robot's location and an observation about the target's position. In the Bayesian network formulation, these two observations taken together are represented by O_R. In our reformulation, these two inputs are used to determine an element from Ω_{S_T} (the robot's best estimate of the target's position), and an element from Ω_{A_R} (the action the robot predicts will have the best expected performance).

The learning problem can be characterized as learning a mapping from Ω_{O_R} to $\Omega_{S_T} \times \Omega_{A_R}$. Since the domain and range are finite, we can build a table where the *index* of the table is Ω_{O_R} and the value stored in each table *entry* consists of an expectation over $\Omega_{S_T} \times \Omega_{A_R}$ for the specified value function. In operation, the robot would make the two observations and

[1] Actually, Howard's method cannot be directly applied to the target localization problem because his method assumes that the states of the world are directly observable. We ignore this to make the point about computational costs.

and everything to do with the way in which the problem is represented and the ways in which knowledge of the domain is exploited to reduce complexity.

Related Work

Probabilistic decision models of the sort explored in this paper are just beginning to see use in planning and control. Agogino and Ramamurthi [1988] describe the use of probabilistic models for controlling machine tools. In their application, the ability to represent the uncertain effects of vibration on the life of cutting tools in a probabilistic decision model provides an advantage over other expert system technologies. Levitt *et al* [1988] describe an approach to implementing object recognition using Bayesian networks that accounts for the cost of sensor movement and inference. Dean *et al* [1990] show how to use Bayesian networks for building maps and reasoning about the costs and benefits of exploration.

References

[Agogino and Ramamurthi, 1988] A. M. Agogino and K. Ramamurthi. Real-time influence diagrams for monitoring and controlling mechanical systems. Technical report, Department of Mechanical Engineering, University of California, Berkeley, 1988.

[Albus, 1975] J. S. Albus. A new approach to manipulator control: The cerebellar model articulation controller (cmac). *Journal of Dynamic Systems, Measurement, and Control*, 97:270–277, 1975.

[Bellman, 1957] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[Cooper *et al.*, 1988] Gregory F. Cooper, Eric J. Horvitz, and David E. Heckerman. A method for temporal probabilistic reasoning. Technical Report KSL-88-30, Stanford Knowledge Systems Laboratory, 1988.

[Dean and Kanazawa, 1988] Thomas Dean and Keiji Kanazawa. Probabilistic temporal reasoning. In *Proceedings AAAI-88*, pages 524–528. AAAI, 1988.

[Dean and Kanazawa, 1989] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[Dean *et al.*, 1988] Thomas L. Dean, R. James Firby, and David P. Miller. Hierarchical planning involving deadlines, travel time and resources. *Computational Intelligence*, 4(4):381–398, 1988.

[Dean *et al.*, 1990] Thomas Dean, Kenneth Basye, Robert Chekaluk, Seungseok Hyun, Moises Lejter, and Margaret Randazza. Coping with uncertainty in a control system for navigation and exploration. In *Proceedings AAAI-90*. AAAI, 1990.

[Feldman and Sproull, 1977] Jerome Feldman and Robert Sproull. Decision theory and artificial intelligence ii: the hungry machine. *Cognitive Science*, 1:158–192, 1977.

[Hanks, 1988] Steve Hanks. Representing and computing temporally scoped beliefs. In *Proceedings AAAI-88*, pages 501–505. AAAI, 1988.

[Heckerman *et al.*, 1989] David E. Heckerman, John S. Breese, and Eric J. Horvitz. The compilation of decision models. In *UW89*, pages 162–173, 1989.

[Howard, 1960] Ronald A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, Massachusetts, 1960.

[Kanazawa and Dean, 1989] Keiji Kanazawa and Thomas Dean. A model for projection and action. In *Proceedings IJCAI 11*, pages 985–990. IJCAI, 1989.

[Langlotz *et al.*, 1987] Curtis P. Langlotz, Lawrence M. Fagan, Samson W. Tu, Branimir I. Sikic, and Edward H. Shortliffe. A therapy planning architecture that combines decision theory and artificial intelligence techniques. *Computers and Biomedical Research*, 20:279–303, 1987.

[Levitt *et al.*, 1988] Tod Levitt, Thomas Binford, Gil Ettinger, and Patrice Gelband. Utility-based control for computer vision. In *Proceedings of the 1988 Workshop on Uncertainty in Artificial Intelligence*, 1988.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan-Kaufmann, Los Altos, California, 1988.

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

[Shewchuk and Dean, 1990] John Shewchuk and Thomas Dean. Learning time-varying functions with high input dimensionality. In *Proceedings of Fifth IEEE International Symposium on Intelligent Control*. IEEE, 1990.

[Shewchuk and Viola, 1989] John Shewchuk and Paul Viola. Implementing a learning system for subsumption architectures. Technical Report CS-89-17, Brown University Department of Computer Science, 1989.

[Sutton, 1990] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings 7th International Conference on Machine Learning*, 1990.

[Watkins, 1989] C.J.C.H Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, 1989.

[Weber, 1989] Jay C. Weber. A parallel algorithm for statistical belief refinement and its use in causal reasoning. In *Proceedings IJCAI 11*. IJCAI, 1989.

[Whitehead and Ballard, 1990] Steven D. Whitehead and Dana H. Ballard. Active perception and reinforcement learning. In *Proceedings 7th International Conference on Machine Learning*, 1990.

[Wilkins, 1984] David E. Wilkins. Domain independent planning: Representation and plan generation. *Artificial Intelligence*, 22:269–302, 1984.