Planning Under Uncertainty: Structural Assumptions and Computational Leverage

Craig Boutilier[*]
Dept. of Comp. Science
Univ. of British Columbia
Vancouver, BC V6T 1Z4
Tel. (604) 822-4632
Fax. (604) 822-5485
cebly@cs.ubc.ca

Thomas Dean[†]
Dept. of Comp. Science
Brown University
Providence, RI 02912
Tel. (401) 863-7645
Fax. (401) 863-7657
tld@cs.brown.edu

Steve Hanks[‡]
Dept. of Comp. Sci. and Eng.
Univ. of Washington
Seattle, WA 98195
Tel. (206) 543-4784
Fax. (206) 543-2969
hanks@cs.washington.edu

Abstract

The problem of planning under uncertainty has been addressed by researchers in many different fields, adopting rather different perspectives on the problem. Unfortunately, these researchers are not always aware of the relationships among these different problem formulations, often resulting in confusion and duplicated effort. Many probabilistic planning or decision making problems can be characterized as a class of Markov decision processes that allow for significant compression in representing the underlying system dynamics. It is for this class of problems that we as experts in intensional representations are advantageously positioned to contribute efficient solution methods. This paper provides a general characterization of the representational requirements for this class of problems, and we describe how to achieve computational leverage using representations that make different types of dependency information explicit.

Keywords: decision-theoretic planning, action representation, uncertainty, stochastic domains

Notice: This paper has not already been accepted by and is not currently under review for a journal or another conference. Nor will it be submitted for such consideration during IJCAI's review period.

[*] This research was supported by NSERC Research Grant OGP0121843, and the NCE IRIS-II program Project IC-7.

[†] This work was supported in part by a National Science Foundation Presidential Young Investigator Award IRI-8957601 and by the Air Force and the Advanced Research Projects Agency of the Department of Defense under Contract No. F30602-91-C-0041.

[‡] This work was supported in part by NSF grant IRI–9008670 and in part by a grant from the University of Washington Royalty Research Fund.

Planning Under Uncertainty: Structural Assumptions and Computational Leverage

Craig Boutilier Thomas Dean Steve Hanks

Abstract

The problem of planning under uncertainty has been addressed by researchers in many different fields, adopting rather different perspectives on the problem. Unfortunately, these researchers are not always aware of the relationships among these different problem formulations, often resulting in confusion and duplicated effort. Many probabilistic planning or decision making problems can be characterized as a class of Markov decision processes that allow for significant compression in representing the underlying system dynamics. It is for this class of problems that we as experts in intensional representations are advantageously positioned to contribute efficient solution methods. This paper provides a general characterization of the representational requirements for this class of problems, and we describe how to achieve computational leverage using representations that make different types of dependency information explicit.

1 Introduction

Much attention has been devoted to planning and decision making under uncertainty by researchers in different fields. While much of this research addresses the same basic set of problems, there is an often unfortunate lack of awareness or appreciation of ideas from other fields, mainly due to the fact that the classes of problems being attacked and their underlying assumptions have never been made explicit. There is a pressing need for a clear statement of the fundamental characteristics of probabilistic planning problems, and the various assumptions and methods of existing solution techniques. It is worthwhile to point out why we believe that research in this area, while currently missing some opportunities for synergy and exploiting previous work, is on the right track. In particular, we show how symbolic and graphical representations used in AI planning and reasoning under uncertainty can help to significantly extend the applicability of some powerful techniques.

Roughly speaking, the class of problems we consider are planning problems whose dynamics can be modeled as stochastic processes. These *Markov decision processes* (MDPs) capture the system dynamics in terms of a probability distribution describing the next state of the system given the current state and any choice of action.[1] We are especially interested in that subclass of problems for which the system's state can be characterized by a set of variables, say of size M, and for which the system dynamics can be compactly represented (in space bounded by some polynomial function of M). Since a set of M boolean variables can represent a state space of size 2^M, this requires that the domain exhibit certain regularities, for example, when certain properties of the next state depends only on some small subset of the M variables. Furthermore we are interested in problems whose solution has a similarly compact representation.

The task confronting research on decision-making agents is the transformation of a compact representation of the system dynamics into a (compact) solution, with the time and space requirements of the process being a polynomial function of M. This task can be accomplished in many cases by exploiting *structure* inherent in the specification of the system dynamics and in the *value function* that determines the quality of a policy. Major goals in planning under uncertainty are to understand this class of problems better, to characterize the problems for which efficient transformations are possible, and to develop efficient solution techniques for these cases.

In this paper we make a small step toward achieving the these goals by describing the classes of problems currently being attacked by researchers and identifying the major sources of structure that can be exploited to expedite the planning process. We begin by describing a broad class of problems, starting with a characterization of the basic dynamical model and followed immediately by a characterization of the associated decision problem. We then describe how certain dynamical models can be compactly encoded using using standard methods from artificial intelligence and other areas of computer science. Finally, we show how structure in the dynamical models, made explicit in such representations, can be used to expedite various sorts of inference critical to solving these decision problems.

2 Markov Decision Processes

In this section we present MDPs: a general framework for planning and decision problems. We first focus on capturing the dynamics of the underlying system, then describe how a planning problem can be formulated as that of choosing actions that alter the system dynamics so that certain criteria are satisfied.

2.1 States, transitions, and trajectories

A *state* is a description of the system at a particular time that captures all information relevant to the agent's decision making process. We assume a finite *state space*

[1] Depending on one's background it is common to talk about a decision maker "controlling a dynamical system" or about "an agent acting in the world." We refer to the decision maker as an *agent* which takes *actions* in the interest of controlling the behavior of some *system*. Its choice of actions is called a *policy*.

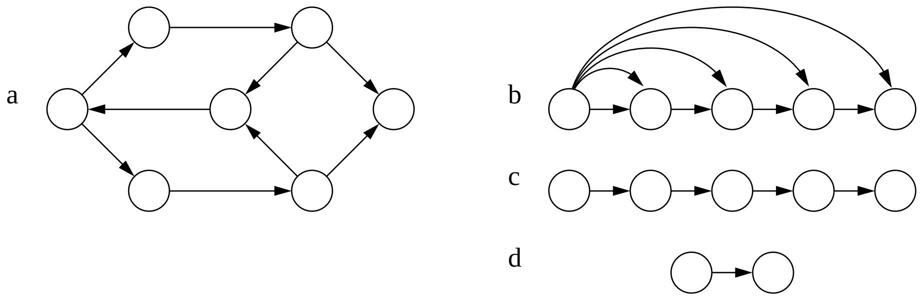


Figure 1: (a) A state-transition diagram; (b) a general stochastic process; (c) a Markov chain; and (d) a stationary Markov chain

$\mathcal{S} = \{s_1, \ldots, s_N\}$ of possible system states. A discrete-time stochastic dynamical system consists of a state space and a probability distribution governing the *next state* of the system given the past states of the system (we assume the agent's course of action is fixed). The sequence of states corresponds to the *stages* of the system.[2] Let $\mathcal{T}$ be the set of all stages.

We generally consider the system's state at some stage t of the process to be a random variable S^t that takes values from the set $\mathcal{S}$. Figure 1(b) shows an alternative view of a discrete-time, stochastic dynamical system in which the nodes are random variables denoting the state at a particular time, and the arcs reflect the dependence of states on previous states. If the next state depends only on the current state (at any fixed stage) then the system is *Markovian*, and can be represented graphically as in Figure 1(c). If the distribution governing the next state is the same for all stages, then the system is *stationary* and can be represented schematically using just two stages as shown in Figure 1(d). A stationary process can also be represented using *state-transition diagram* (Figure 1(a)), where nodes correspond to states and arcs denote possible transitions (labeled with transition probabilities). The size of such a diagram is at least $O(N)$ and at most $O(N^2)$, depending on the number of arcs (possible transitions).

We will usually restrict our attention to discrete-time, finite-state, stochastic dynamical systems with the Markov property; *i.e., Markov chains*. We can represent an T-stage Markov chain using T *transition matrices* of size $N \times N$. Each matrix consists of probabilities p_{ij}, where $p_{ij} = \Pr(S^{t+1} = s_j | S^t = s_i)$. If the process is stationary, one matrix will suffice. Finally, a *trajectory* or *history* through the state space is a sequence of states, $\langle S^0, S^1, \ldots, S^T \rangle$, denoting the evolution of the system for T stages. Given a set of transition matrices, a distribution over starting states induces a distribution over trajectories.

[2] The stage of a system provides a loose notion of time. An action causes a transition from one stage t to the next $t + 1$. Under the assumption that all actions take a single unit of time, this view is equivalent to saying that an action "moves" the system from *time* t to time $t + 1$. This assumption is not required for any of what follows.

2.2 Actions, policies, and value functions

We assume that our system can be controlled (to some extent) by an agent. The agent's *action* choices determine particular transition probabilities, thus influencing the distribution over histories. The agent therefore must choose a course of action that produces a "good" distribution over histories according to some measure.

We assume that an agent has available to it a finite set of *actions* $\mathcal{A} = \{a_1, \ldots, a_K\}$ that influence the state of the system probabilistically. Formally, an action a_k maps each state s_i into a distribution over $\mathcal{S}$ that characterizes the possible outcomes of that action. An action can be represented by a set of T transition matrices with entries $p_{ij}^{k,t} = \Pr(S^{t+1} = s_j | S^t = s_i, A^t = a_k)$, the probability that the system will move to state s_j when a_k is executed in state s_i at stage t ($1 \leq t \leq T$). It is typically assumed that the action's effects are independent of the stage at which it is executed. In this case an action can be described by a *single* N by N transition matrix p_{ij}^k.

Although the effects of an action can depend on any aspect of the prevailing state, often the agent cannot *observe* every aspect of the state. Since its decisions can be based only on what it observes, we model the (potentially restricted) observational or sensing capabilities of an agent by introducing a finite set of *observations* $\mathcal{O} = \{o_1, \ldots, o_H\}$. The agent receives an observation prior to its action choice at each stage t, denoted by the random variable O^t. We let $\Pr(O^{t+1} = o_h | S^t = s_i, A^t = a_k, S^{t+1} = s_j)$ be the probability that the agent observes o_h at stage $t + 1$ given that it performs a_k in state s_i and ends up in state s_j.

This model allows a wide variety of assumptions about the agent's sensing capabilities. At one extreme are *fully observable MDPs* (FOMPDs), in which the agent knows exactly what state it is in at each stage t. We model this case by letting $\mathcal{O} = \mathcal{S}$ and $\Pr(o_h | s_i, a_k, s_j) = 1$ iff $o_h = s_j$. At the other extreme are *non-observable* systems in which the agent receives *no* information about the system's state during execution. We can model this case by letting $\mathcal{O} = \{o\}$; the same observation is reported at each stage, revealing no information about the state (*i.e.*, $\Pr(s_j | s_i, a_k, o) = \Pr(s_j | s_i, a_k)$). We can also model intermediate cases in which the agent receives incomplete or noisy information about the system state (*i.e.*, *partially observable* MDPs, or POMDPs).

We extend the notion of history to account for actions and observables in the obvious way: a history is a sequence of tuples

$$\langle \langle S^0, O^0, A^0 \rangle, \langle S^1, O^1, A^1 \rangle, \ldots \langle S^T, O^T, A^T \rangle \rangle$$

summarizing the state, observation and action for the duration of the system's execution. The *observable* history at stage t corresponds to the finite sequence $\langle \langle O^0, A^0 \rangle, \ldots, \langle O^{t-1}, A^{t-1} \rangle \rangle$, where O^0 is the observation of the initial state. The observable history at stage t constitutes all the information available to the agent in deciding how to act at stage t.

The general decision problem can be described as building a *policy* π, which maps the observable history at stage t to an action: $\pi : \mathcal{O} \rightarrow \mathcal{A}$. Extreme assumptions about observability simplifies the structure of the policy somewhat. In the fully observable case a policy depends only on the (current) state and the stage: $\pi : \mathcal{S} \times \mathcal{T} \to \mathcal{A}$. In the non-observable case the agent must act based only on knowledge of its previous actions (and stage); this form of policy corresponds to a linear unconditional sequence of actions (a straight-line plan) $\langle A^1, A^2, \ldots, A^T \rangle$.

Clearly, a policy induces a distribution $\text{Pr}(h|\pi)$ over the set of histories $\mathcal{H}$. In order to distinguish certain policies as preferred, we define a *value* function $\text{V}(\cdot)$ that maps histories to the reals, $\text{V} : \mathcal{H} \to \mathbf{R}$; the agent prefers history h to h' just in case $\text{V}(h) > \text{V}(h')$. We then define the expected value of a policy to be:

$$\mathbf{EV}(\pi) = \sum_{h \in \mathcal{H}} \text{Pr}(h|\pi)\text{V}(h)$$

We will consider planning problems that vary in two dimensions: the horizon over which the value function is evaluated; and the criterion that determines whether a particular policy should be considered a solution to a problem.

Finite-horizon problems evaluate the agent's performance over a fixed, finite number of stages M. An infinite horizon problem, on the other hand, considers performance over an arbitrary period of time. We will also consider *indefinite-horizon* problems in which the policy is executed until some terminating or goal state is reached. An indefinite-horizon problem can always be recast as an infinite-horizon problem, but generally cannot be represented using a fixed finite horizon.[3]

We will also distinguish *optimization* problems from *satisficing* problems. In the former case the agent seeks a policy that maximizes expected value, whereas in the latter it seeks a policy π such that $\mathbf{EV}(\pi) > \tau$ for some fixed threshold $\tau \in \mathbf{R}$.

This framework for posing decision problems is quite general, subsuming a number problems commonly addressed in the decision-making and planning literature, such as classical planning and FOMDPs. We demonstrate this below.

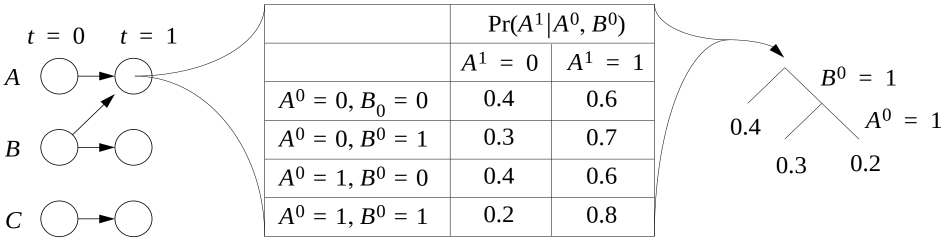
3 Structural Assumptions and Structured Representations

The generality of the framework for posing planning problems described in the previous section comes at a high price in terms space (for storing the transition matrices and the policy itself), time (required to generate a solution policy), and ease of specification. By examining assumptions that admit *structure* in the representation of state, actions, value functions, or policies, we can identify classes of problems that can potentially be solved effectively.

3.1 Structure in the state

We will begin by examining *structured states*, or systems whose state can be described using a finite set of *state variables* (or fluents) whose values change over time. If there are M such (say, boolean) variables then the size of the state space is $|\mathcal{S}| = N = 2^M$. For large M, specifying or representing the dynamics explicitly

[3] We consider particular forms for value functions below.



	$\Pr(A^1 \mid A^0, B^0)$	
	$A^1 = 0$	$A^1 = 1$
$A^0 = 0, B_0 = 0$	0.4	0.6
$A^0 = 0, B^0 = 1$	0.3	0.7
$A^0 = 1, B^0 = 0$	0.4	0.6
$A^0 = 1, B^0 = 1$	0.2	0.8

Figure 2: Two-stage Bayes network (left) with conditional probability distribution (center) for one node and decision-tree (right) for that distribution

using state-transition diagrams or $N \times N$ matrices is impractical. In the following, we define a class of problems in which the dynamics can be represented in $O(M)$ space. We begin by considering how to represent Markov chains compactly and then consider incorporating actions and representing other aspects of the value function.

We say the state space is *flat* if it is specified using one state variable (this variable is denoted S as in the general model, taking values from $\mathcal{S}$). The state space is *factored* if there is more than one state variable. A state is then any possible assignment of values to these variables. Letting X_i represent the ith state variable and Ω_{X_i} its (finite set of) possible values, we have $\mathcal{S} = \Omega_S = \prod_{i=1}^{M} \Omega_{X_i}$. We let X_i^t be the random variable representing the value of the ith state variable at stage t.

Bayesian networks provide a compact representation for factored state spaces. A Bayes net is a directed acyclic graph with vertices corresponding to random variables and edges indicating informational dependencies among the variables. To *quantify* a Bayes net we supply a conditional probability distribution for each variable given its parents and a marginal distribution for those vertices that have no parents. Figures 1(b)-(d) are special cases of Bayes nets, called "temporal" networks, in which the conditional distributions are state-transition probabilities and the marginal distributions are initial state distributions.

A *two-stage temporal Bayes network* (2TBN) is a Bayes net with two sets of variables $U^t = \{X_1^t, \ldots, X_M^t\}$ and $U^{t+1} = \{X_1^{t+1}, \ldots, X_M^{t+1}\}$ with each arc either (i) from a variable in U^t to a variable in U^{t+1} or (ii) between variables in U^{t+1}. 2TBNs afford us a compact representation for large state spaces, if they are factored and the dependencies induced by the actions (represented by arcs from one stage to the next) are relatively few. If there are M boolean state variables, each with no more than L parents in the 2TBN, then the size of the factored representation is $O(M2^L)$. This is due to the fact that we require for each variable X_i^t its probability given the any of the 2^L possible value assignments to its L parents. This conditional distribution is typically represented in the tabular form illustrated at the center of Figure 2. This representation is said to be locally exponential (in L) but globally linear (in M). At one extreme, if X_i depends only on itself, and the system can be decomposed into M independent processes. At the other extreme the case in which X_i depends on all the $X_1, \ldots, X_M$ corresponds to a flat state space.

The Bayes net representation is equivalent in expressive power to the general *stationary* transition matrix model: every 2TBN has a natural interpretation as a stationary Markov chain (assuming an initial distribution); and likewise every stationary Markov chain can be trivially represented as a 2TBN (see, *e.g.*, Figure 1(b)). However, if the state space of a Markov chain is factored with certain independencies among variables, then a more concise 2TBN can be built that reflects these independencies, allowing for compact representation of structured stationary chains.

A *simple* 2TBN is a 2TBN with no arcs between state variables in the second stage.[4] Note that *not* every Markov chain in factored form can be represented as a simple 2TBN. In particular, any Markov chain with correlations among state variables cannot be represented in a fully factored form. We can construct a *simple* 2TBN for a stationary chain by collapsing variables that exhibit intra-stage correlation, forming a conjunction or "joint" variable; this gives up some of the factorization of the state space, and incurs a local combinatorial explosion in the representation.

The use of 2TBNs for Markov chains only captures *variable independence*, or independence among variables without regard for specific values they might take. *Propositional independence*, or independence of specific variable *assignments* will only become apparent when examining the specific distribution in the conditional probability table for a given node. We can graphically represent propositional independence by representing the conditional distribution using a decision tree instead of a table, allowing an even more concise description of an action's effects.[5] The decision tree on the right of Figure 2 is equivalent to the illustrated table, but makes explicit the fact that the transition probability for A^1 is independent of the value of A^0 given that $B^0 = 0$.

3.2 Structure in actions and value function

As in the general model, we must represent the possible choices facing an agent and the value of various histories, in addition to the system dynamics. We expect, just as the representation of states can be factored and expressed compactly, that actions and value functions will also exhibit regularities that can be exploited.

In the general model, the effects of an action are represented with a transition matrix.[6] If the effects of an action exhibit structure (*e.g.*, if that action has no effect on certain state variables), a 2TBN can be used in the obvious way to represent that action. The decision problem can then be represented using one 2TBN for each action. However, we can exploit regularities among different actions by using one 2TBN in which all actions are described. In particular, we represent actions using an *action variable*, whose value (an action) is "chosen" by the agent at any stage. The conditional distribution for a variable then depends not only on its state-

[4] Such arcs are generally referred to as *synchronic* or *domain* constraints or *ramifications*.

[5] More generally, acyclic decision graphs might be used.

[6] We assume for the remainder of this section a stationary dynamics.

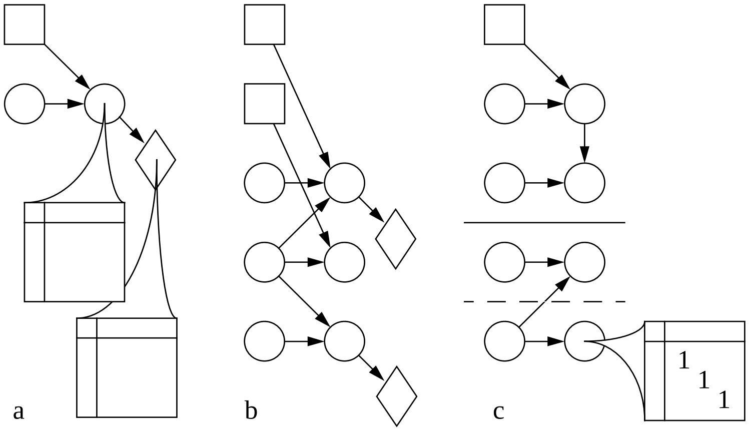


Figure 3: (a) Actions and costs in an influence diagram; (b) action parameters and cost attributes; (c) independent and unaffected state variables

variable parents, but also on the particular action choice. Using decision trees, the representation need not blow up (compared to one 2TBN per action) through the introduction of a new multivalued parent variable; and savings are possible when actions share similar effects on given variables.

It is common to make explicit the actions (and value, as we describe below) using a generalization of Bayes nets called *influence diagrams* [13]. Figure 3(a) shows the influence diagram equivalent of a 2TBN; action variables (or *decision nodes*) are depicted as boxes, cost functions as diamonds, and state variables as circles.[7] There are additional opportunities for representational economy that arise in 2TBNs; often an action can be factored or parameterized, with the effect on different variables depending only on certain factors (see Figure 3(b)).

A more extreme case of a variable being independent of action is illustrated in the use of the STRIPS assumption and implicit *frame axioms*. In models with boolean variables, we can describe states using logical propositions. A variant of the decision tree representation of actions are *probabilistic state space operators* (PSOs) — an extension of the classical STRIPS operators [10]. A PSO α is a set of triples of the form $\langle \phi, \rho, \omega \rangle$ where ϕ is a set of propositions (preconditions) that describe a subset of $\mathcal{S}$, ρ is a probability, and ω is a set of propositions (postconditions) describing another subset of $\mathcal{S}$. Semantically, if ϕ is satisfied just prior to α, then with probability ρ the postconditions in ω are satisfied immediately following α. If a proposition is not included in ϕ, then it is assumed not to affect the outcome of α; if a proposition is not included in ω, then it is assumed to be unchanged by α.[8] For

[7] Actions may have incoming arcs (as we describe below) denoting available observations; but for now we assume the agent is aware of the entire state (since we are assuming stationarity, previous history is irrelevant).

[8] The table in Figure 3(c) illustrates such a "persistence" distribution.

example, given the following representation for α

$$\alpha = \{\langle\{P\}, 1, \emptyset\rangle, \langle\{\neg P\}, 0.2, \{P\}\rangle, \langle\{\neg P\}, 0.8, \{\neg P\}\rangle\}$$

if P is true prior to α, nothing is changed following α (all variables *persist*); but if P is false, then 20% of the time P becomes true and 80% of the time P remains false. If the 2TBN in Figure 2 represents some action α, taking value 0 to be false and 1 to be true, it can be written as the following PSO (assuming B, C persist):

$$\alpha = \{\langle\{A, B\}, 0.8, \{A\}\rangle, \langle\{\neg A, B\}, 0.7, \{A\}\rangle, \langle\{\neg B\}, 0.6, \{A\}\rangle\}$$

Probabilistic state-space operators have certain advantages over other representations. First, the conditions under which an action has distinct effects (*i.e.*, gives rise to different *joint* distributions) can be represented using arbitrary propositions. Second, the effects of an action on distinct variables under a given set of conditions can be encoded together, thus eliminating the need for a redundant partitioning of the state space for every affected variable. One drawback of this representation is the need to specify mutually exclusive propositional partitions for each condition under which the action induces a distinct *joint* distribution. Network representations, on the other hand, only require partitions for each distinct variable (though see [4] for factored PSOs).

Structure in the Value Function: Value functions, in the general case arbitrary functions of histories, also typically exhibit useful structure. A *time-separable* value function V is one in which the value assigned to a history is some function (generally a sum or product) of sub-value functions applied to each stage of the process.[9] Furthermore, the sub-value functions are typically stationary, and thus can be captured in a 2TBN (Figure 3). Although some useful value functions (*e.g.*, temporal deadlines and maintenance intervals) cannot be represented effectively under such assumptions, the advantage of separability is that the performance function **EV**$(\cdot)$ can be represented using a small number of parameters (the size required to represent V depends on the size of the state space and not the length of the process).

Another common simplification of the value function is the notion of a *goal*, which is some subset of the state space identified as valuable. An agent that derives value *only* from achieving a goal can be modeled using a separable value function in which the only goal states are assigned a non-zero reward. A richer model might allow goal states with (positive) rewards and non-goal states with (negative) costs.

4 Computational Leverage

We have seen how structural assumptions about the domain admit natural parameterizations and compact representations for states, actions, and value functions. But

[9] In an infinite-horizon problem sums or products of sub-values is usually not be bounded; in this case, value per stage is often *discounted*, or the limiting average per stage is used to judge quality.

we also want to use such representations to make explicit these structural properties in the hope that they can be exploited by solution algorithms. In this section, we survey some existing solution techniques, showing how their effectiveness depends on structural features of the domain, and suggest ways in which these solution techniques might be combined to produce even more effective algorithms.

4.1 Dynamic programming

Dynamic programming (DP) is the most common technique for solving MDP problems. Originally proposed in [1], the DP technique is based on the following "principle of optimality." If $\mathbf{EV}_t^*(s)$ is the optimal expected value for the t-stage policy given that the system is in state $s \in \mathcal{S}$, then

$$\mathbf{EV}_{t+1}^*(s) = \mathrm{C}(s) + \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \Pr(s'|s,a)\mathbf{EV}_t^*(s) \qquad (1)$$

In other words, an optimal solution to the $(t+1)$-stage problem can be expressed in terms of the optimal solution to the t-stage problem. DP implements this recurrence relation, and computes the optimal policy for a fixed-horizon T-stage policy (for some constant T) in time polynomial in $|\mathcal{S}|$, $|\mathcal{A}|$ and T. The space required to store the policy is $O(T|\mathcal{S}|)$.

Note the two crucial simplifying assumptions made in this formulation of the problem: full observability and a separable value function. Full observability is a particularly powerful assumption, allowing a policy to be expressed as a mapping from $\mathcal{S} \times \mathcal{T}$ to $\mathcal{A}$; furthermore the Markov assumption for $\mathcal{S}$ guarantees that the optimal action choice at stage t depends only on S^t. The separability of the value function is crucial to the recursive nature of the problem, allowing the optimal $(t+1)$-stage policy to be expressed as a function of the current state and the optimal t-stage value.

The principle of optimality can be applied to infinite-horizon problems as well. These include *expected average-per-stage reward* problems and classic *expected total discounted reward* problems [1, 12], where the value function is the discounted sum of rewards incurred at every stage of execution:

$$\mathrm{V}(h) = \sum_{t=0}^{\infty} \gamma^t \mathrm{C}(S^t, A^t, S^{t+1})$$

where γ is a fixed *discount rate* ($0 < \gamma < 1$). Optimal policies for these problems, assuming FOMDPs, are stationary, therefore requiring only $O(|\mathcal{S}|)$ space.

DP techniques can be applied in partially-observable settings as well [23], but the problem is that the state space becomes the set of all *probability distributions* over $\mathcal{S}$, which can, in the worst case, grow exponentially with the number of stages in the problem. Using DP to solve POMDPs (with an additive discounted value function) is currently practical only for very small problems [17].

Although DP techniques were developed using flat state-space and action representations, the same basic technique can be applied to factored representations,

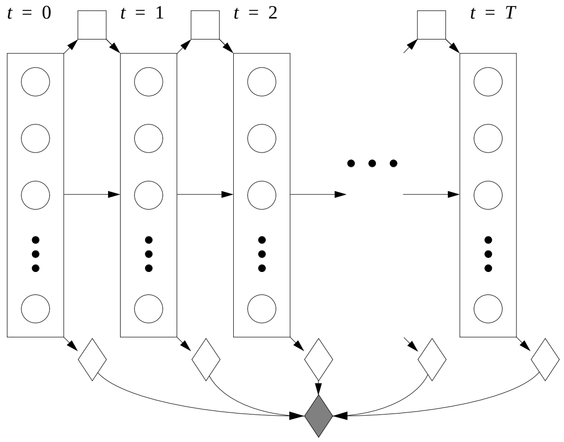


Figure 4: A temporal influence diagram

exploiting problem structure. Figure 4 illustrates a general influence diagram (in contrast to the 2TBN structures in Figure 3). The highlighted value node represents the value of the policy, and is a function of the sub-value nodes associated with each stage. The arrows into the action variables capture the available information when the decisions at the corresponding stages are made.

This influence diagram represents a finite-horizon, fully observable[10] problem with a separable value function — the horizon is inherently fixed by the number of stages in the graph. DP techniques can be employed to solve problems of this sort [24]: while the same basic principle of optimality is used, computational leverage can be gained by algorithms that use the structured representation to identify relevant or irrelevant variables at various stages of the process, given decisions at later stages [24, 5]. As such, groups of states, "indistinguishable" in relevant details, are treated as a single state for computational purposes.

4.2 Backchaining

Most AI planning algorithms are based on the idea of *backchaining*: the algorithm identifies a desired state, chooses an operator that would effect that state, then adopts the subproblem of trying to enable that chosen operator. Classical planning algorithms have implemented this approach using a variety of algorithms, all involving a *goal state* and symbolic state-space operators.

Classical planning problems can be posed in terms of an indefinite-horizon non-observable system with deterministic dynamics: there is a single initial state and for every action k and state s_i there is some state s_j such that $p_{ij}^k = 1$. Therefore a policy (straight-line plan) π defines a single history, and the value function assigns this history 1 if the final state in the history is a goal state and 0 otherwise.

[10] That is, the complete state at any stage t is known when making a decision at that stage. Partially observable problems are captured by highlighting observable variables or using *information arcs* into decision nodes.

Though most classical planning problems have been cast in deterministic terms, the Buridan and C-Buridan planners [16, 9] reformulate the problem in probabilistically: given a distribution over states, a set of actions, a goal expression, and a probability threshold τ, generate a sequence of actions that when executed will leave the system in a goal state with probability at least τ. This problem can be recast as a POMDP with indefinite horizon, requiring a policy with expected value at least τ.[11]

The Buridan planners use classical backchaining techniques to solve this problem, making use of two key structural properties of the problem: goal states instead of arbitrary value functions means that these are the *only* states with reward; and state-space operators impose structure on the system dynamics. Backchaining is essentially the matching of desired world states with appropriate actions. A key advantage of backchaining is that the agent needs to consider only those aspects of the state and those actions are relevant *to the immediate problem*. If a state variable or action is irrelevant to achieving the current goal, the algorithm need not reason about it at all; such irrelevance can be identified in the action representation. In a sense, these planners (as well as classical planners) backchain over propositions, or structured clusters of states, implicitly grouping together states that are identical in the relevant details. The computational complexity of building a plan need not grow as irrelevant variables or actions are added to the domain.

Another key assumption is the existence of goal states. This is in some sense the problem feature that obviates the need for DP and makes backchaining feasible. In a general MDP, *any* state can contribute to the policy's value. DP over a finite horizon is *undirected*: since all (reachable) states have potential impact, all must be considered when constructing a policy. DP is simply an efficient way to consider all states; but even if structured system dynamics *reduce* the "effective state space" [24, 5], all groups of states must be visited. In a goal-based setting, no value is accrued for states visited in the process of achieving the goal nor for anything that happens after the goal is achieved.[12] An optimal plan must reach a goal state. Thus the search for a plan (and associated trajectory through the state space) can be *directed* toward goal states; backchaining exploits and depends crucially on this directedness.

4.3 Abstraction and aggregation

By exploiting structured representations, both DP and backchaining can be used in a manner that groups together various states that are differ only in irrelevant features. This type of *aggregation* is *adaptive*: the variables identified as relevant at one stage of the problem (or the computation) may be different from those at another

[11] The Buridan planner solves a *non-observable* process, and generates straight-line plans; C-Buridan extends the action representation to allow observational actions and the plan representation allows actions to be executed contingent on the result of observations.

[12] Though see [26] for a system that uses backchaining in the context of a value function that also takes into account the costs and benefits of alternative plans that achieve the goal.

stage. For example, while only the goal proposition is initially relevant in a classical planning problem, once a subgoal has been identified, *its* propositions are the ones deemed relevant. Planning algorithms and influence diagram solution techniques implicitly perform such classifications.[13]

One can also perform nonadaptive aggregation or *abstraction* of a planning problem; that is, we can identify relevant variables *prior* to using a planning or policy construction algorithm, and construct an abstract state space consisting of *aggregate states*. Such aggregate states typically consist of groups of states that differ only on irrelevant features. A crucial aspect of such abstraction mechanisms is the ability to construct an abstract decision problem whose dynamics and value function is suitable for the abstract state space. Once again the structured representation of actions and value functions often allow one to determine the relevance of variables and action prior to planning. In classical hierarchical planning [20], state-space operators suggest the criticality of various propositions, allowing one to plan for the most important propositions. In MDPs, 2TBNs with value nodes or PSOs can be used to identify the relative contribution variables make to overall value [4].

By ignoring variables, we generally cannot guarantee the discovery of an optimal (or even satisficing) policy, unless these variables are “strictly” irrelevant. In the classical setting, an abstract plan is used to guide the search for a concrete plan [20] with this guarantee. In general MDPs, nonoptimal plans will have some value; by identifying and eliminating variables with relatively little contribution to value, satisficing policies are thus feasible [4]. Ultimately, more adaptive, nonuniform *prior* aggregation techniques should offer the advantages of both forms of aggregation.

5 Related Work

The general model we have presented here and the particular representations we described draw tremendously from existing work. Space limitations prevent the comprehensive discussion of related work that is warranted; but we briefly mention some of the especially relevant research.

Bertsekas [3] and Puterman [19] provide excellent and extensive coverage of Markov decision processes and dynamic programming. Dean and Wellman [8] present the view of planning problems in terms of expected value over histories, while Dean *et al.* [6] propose MDPs as a model for planning in stochastic domains.

Dean and Kanazawa [7, 14] introduce the notion of 2TBNs and probabilistic action networks based on influence diagrams. Luenberger [18] provides a definition and discussion of separable value functions. Tatman and Shachter [24] show how separable value functions are incorporated in influence diagrams and their relation to dynamic programming. Boutilier *et al.* [5] describe similar ideas using 2TBNs with decision trees for infinite horizon problems and compact policy representations. Shachter and Peot [22] describe how value nodes can be replaced by chance nodes

[13] See [2] for adaptive aggregation in flat states spaces.

to solve influence diagrams using probabilistic inference; see Savage [21] on the foundational relationship between utilities and probabilities.

Wellman [25] describes a STRIPS assumption for planning under uncertainty, while Hanks and McDermott [11] introduce probabilistic state-space operators. Kushmerick *et al.* [16] propose a method for open-loop probabilistic planning using PSOs in nonobservable domains, while Draper *et al.* [9] address closed-loop probabilistic planning in partially-observable settings (see also [23, 17] for the classical definition of POMDPs).

Sacerdoti [20] uses abstraction for planning in classical settings, while Knoblock [15] addresses the question of generating abstractions using STRIPS action descriptions. Boutilier and Dearden [4] propose a nonadaptive abstraction method for MDPs that exploits factored representations of actions and value functions.

6 Conclusion

Research in decision-theoretic planning and planning under uncertainty has taken many different directions, and each line of research adopted a particular set of assumptions about the underlying problem and about the target problem domain. Frequently, neither the assumptions, nor the relationship between assumptions and proposed representation and solution techniques, have been made clear.

We have presented a general model for the specification of planning problems using Markov decision processes, described several representations for factored state spaces, action spaces and value functions, and shown how these can compactly represent an MDP that exhibits structure. We have argued that one of the main goals in planning under uncertainty is to exploit these compact representations by developing algorithms that efficiently transform compact problem specifications into compact solution representations when problems exhibit the required structure. By describing how existing solution techniques use particular representations we are able to identify the key structural assumptions underlying their success. Furthermore, with these assumptions in hand, we expect that even more powerful methods for planning under uncertainty can be developed, leading to the discovery of new classes of problems that can be solved effectively.

References

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

[2] D. P. Bertsekas and D. A. Castanon. Adaptive aggregation for infinite horizon dynamic programming. *IEEE Trans. on Aut. Control*, 34(6):589–598, 1989.

[3] D. P. Bertsekas. *Dynamic Programming*. Prentice-Hall, 1987.

[4] C. Boutilier and R. Dearden. Using abstractions for decision theoretic planning with time constraints. *AAAI-94*. Seattle, 1994.

[5] Craig B., R. Dearden, and M. Goldszmidt. Exploiting structure in policy construction. To appear AAAI Spr. Symp. on Extending Theories of Action, 1995.

[6] T. Dean, L. P. Kaelbling, J. Kirman, and A. Nicholson. Planning with deadlines in stochastic domains. *AAAI-93*, pp.574–579. Washington, DC, 1993.

[7] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Comp. Intel.*, 5(3):142–150, 1989.

[8] T. Dean and M. Wellman. *Planning and Control*. Morgan Kaufmann, 1991.

[9] D. Draper, S. Hanks, and D. Weld. Probabilistic planning with information gathering and contingent execution. *2nd Intl. Conf. AI Planning Systems*, 1994.

[10] R. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Art. Intel.*, 2:189–208, 1971.

[11] S. Hanks and D. V. McDermott. Modeling a dynamic and uncertain world i: Symbolic and probabilistic reasoning about change. *Art. Intel.*, 1994.

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

[13] R. A. Howard and J. E. Matheson. Influence diagrams. In R. A. Howard, J. E. Matheson, editors, *The Principles and Applications of Decision Analysis*. Strategic Decisions Group, Menlo Park, 1984.

[14] K. Kanazawa and T. Dean. A model for projection and action. *IJCAI-89*, pp.985–990. Detroit, 1989.

[15] C. A. Knoblock. *Generating Abstraction Hierarchies: An Automated Approach to Reducing Search in Planning*. Kluwer, 1993.

[16] N. Kushmerick, S. Hanks, and D. Weld. An algorithm for probabilistic planning. *AAAI-94*. Seattle, 1994.

[17] W. S. Lovejoy. A survey of algorithmic methods for partially observed markov decision processes. *Annals of Op. Res.*, 28:47–66, 1991.

[18] D. G. Luenberger. *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, 1973.

[19] M. L. Puterman. *Markov Decision Processes*. Wiley, New York, 1994.

[20] E. D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Art. Intel.*, 5:115–135, 1974.

[21] L. J. Savage. *The Foundations of Statistics*. Dover, 1972.

[22] R. Shachter and M. Peot. Decision making using probabilistic inference models. *UAI-92*, Stanford, 1992.

[23] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Op. Res.*, 21:1071–1088, 1973.

[24] J. A. Tatman and R. D. Shachter. Dynamic programming and influence diagrams. *IEEE Trans. on Sys., Man, and Cyber.*, 20(2):365–379, 1990.

[25] M. P. Wellman. The STRIPS assumption for planning under uncertainty. *AAAI-90*, pp.198–203. Boston, 1990.

[26] M. Williamson and S. Hanks. Optimal planning with a goal-directed utility model. *2nd Intl. Conf. on AI Planning Systems*, 1994.