Decision-theoretic Refinement Planning in Medical Decision Making:

Management of Acute Deep Venous Thrombosis

PETER HADDAWY, PhD, ANHAI DOAN, MS, CHARLES E. KAHN, Jr., MD

Decision-theoretic refinement planning is a new technique for finding optimal courses of action. The authors sought to determine whether this technique could **identify** optimal strategies for medical diagnosis and therapy. An existing model of acute deep venous thrombosis of the lower extremities was encoded for analysis by the decision-theoretic refinement planning system (DRIPS). The encoding represented 6,206 possible plans. The DRIPS planner **used artificial intelligence techniques to eliminate 5,150 plans (63%) from consideration without examining them explicitly. The DRIPS system identified the five strategies that minimized cost and mortality. The authors conclude that decision-theoretic planning is useful for examining large medical-decision problems. *Key words:* decision-theoretic refinement planning; diagnosis; treatment; DRIPS; acute deep venous thrombosis; artificial intelligence. (Med** Decis Making 1996;16:315-325)

To solve complex problems, decision analysts often must evaluate numerous potential strategies by constructing decision trees by hand. This manual process has several drawbacks. First, because constructing the trees is labor-intensive, analysts may limit the number of strategies to be explored and incorrectly may exclude potentially useful strategies.[1,2] Second; decision trees for large problems may be complex, and manual construction of complex decision trees is prone to error. Third, decision **trees** have a very low degree of modularity, making them difficult to alter. This is a serious drawback since models are typically developed in an iterative fashion.

The decision-theoretic refinement planning system **(DRIPS)** automates the process of decision-tree construction and evaluation. The system provides a modular representation and uses abstraction techniques to efficiently explore a large number of possible strategies.[3] It finds the strategy or strategies that maximize the expected value of a user-defined utility function or a set of outcome attributes. With these capabilities, DRIPS provides a mechanism to identify optimal strategies involving diagnosis and treatment. To study the applicability of this technique to medical decision making, we applied DRIPS to data from a published analysis of management of acute lower-extremity deep venous thrombosis (DVT), in which 24 strategies (out of a space of 6,206 possible strategies1 were evaluated.4 To evaluate the efficiency of the technique, we compared its performance on this problem with that of a standard branch-and-bound decision-tree algorithm.

In this article, we first describe the model of DVT that serves as a domain in which to demonstrate our technique. We then describe the DRIPS planner, and provide details of the action representation, the abstraction techniques, and the planning algorithm. We finish by presenting the results of the analysis of the DVT domain produced by DRIPS and comparing the performance of DRIPS with that of conventional decision trees.

Received January 10, 1995, from the Decision Systems and Artificial Intelligence Laboratory, Department of Electrical Engineering and Computer Science, University of Wisconsin-Milwaukee (PH, AD, CEK); and the Section of Information and Decision Sciences, Department of Radiology Medical College of Wisconsin (PH, CEK), Milwaukee, Wisconsin. Revision accepted for publication February 15, 1996. Presented in part at the 16th annual meeting of the Society for Medical Decision Making, October 18,1994. Supported in part by National Science Foundation grants #IRI-9207262 and #IRI-9509165 (PH) and by the 1993 American Roentgen Ray Society Scholarship (CEK). This work was facilitated by a grant from the National Library of Medicine (USPHS GO8 LM05705) to the Medical College of Wisconsin for integrated advanced information management systems (IAIMS) planning.

Address correspondence and reprint requests to Dr. Haddawy: Department of Electrical Engineering and Computer Science, University of Wisconsin-Milwaukee, PO Box 784, Milwaukee, WI 53201. e-mail: (haddawy@cs.uwm.edu).

*The complete domain model (electronic or hard copy) is available from the first author (PH) upon request.

†Action names are displayed in bold sans serif type when they refer to Boolean tree or DRIPS action **representations.**

Acute Deep Venous Thrombosis

To evaluate the applicability of **DRIPS** to medical decision making, we constructed a model for diagnosis and treatment of acute deep venous thrombosis of the lower extremities. Appropriate management of patients with suspected DVT remains an important and complex clinical problem. The clinical findings of DVT do not permit diagnosis with certainty.[5,6] Unchecked, lower-extremity DVT can progress to pulmonary embolism (PE), a condition that entails significant morbidity and mortality. Anticoagulation therapy for DVT is expensive and carries the risk of severe hemorrhage. Even diagnostic procedures such as venography entail risks such as contrast reaction and iatrogenic DVT. The objective of applying decision analysis to this problem is to determine the optimal strategies for testing and treating a patient suspected of having DVT.

Our model was based on data from an article that compared 24 different management strategies.[4] We chose this article because it contained explicit probability and cost data, and because we hoped to show the advantage of our technique over the manual construction of decision trees. To encompass all of the strategies described in the original model, our model incorporated up to four tests, with a maximum of three seven-day waiting periods between tests. The test procedures included contrast venography (Veno) and two noninvasive tests (NITs): impedance plethysmography (IPG) and real-time ultrasonography (RUS). Treatment, which consisted solely of anticoagulation therapy, included unconditional actions (e.g., Treat all) and conditional actions (e.g., Treat if thigh DVT seen on venography).* Although the term "thigh DVT" is not defined in the reference model, we take it to mean thrombosis of the superficial femoral vein, deep femoral vein, and/or common femoral vein.

Action Representation

The standard method of representing strategies and computing their outcomes in clinical decision analysis is to use decision trees.' Since explicitly creating the trees is labor-intensive, researchers in clinical decision analysis have come up with a number of representation schemes that help compactly specify large decision trees." One such scheme is the ***subtree*** representation.[7] When specifying a decision tree, the repetitive tree fragments are replaced with pointers to a common subtree. Another representation scheme that is used in the same manner as the subtree representation but is more expressive is the Boolean ***node*** representation.' A Boolean node (or rather a Boolean tree) is a tree representing a conditional statement that describes under which conditions which tree fragment is to be used. Figure 1, a.l-a.3, shows examples of subtrees and Boolean trees. The introduction of Boolean trees and subtrees significantly simplifies the representation of decision trees.

Extensive research in artificial intelligence on modeling actions under uncertainty has produced a representation similar to Boolean trees that we call the STRIPS-*style* ***probabilistic action model,'*** which compactly represents strategies (thus decision trees), yet is semantically clear and modular. Variants of this representation are the underlying representations of many current artificial intelligence planning systems.[3,10,11]

To represent strategies using the STRIPS-style probabilistic action model, we first identify the set of all actions that can be performed by the decision maker. There will be a subset of this action set that is the smallest set in the sense that any action that the decision maker can perform can be constructed in a simple way from the actions in the set using conditional statements. We call the actions in this smallest set the ***basic actions*** and the remaining actions the ***composite actions.*** For example, in the DVT domain the eight basic actions are: IPG, RUS, venography, no test, treatment, no treatment, wait between tests, and no wait between tests; a composite action is "if venography shows thigh DVT then treat; otherwise do not treat."

Next, we depict the changes produced after taking a basic action with a Boolean tre,e. In this Boolean tree each condition (logical expression1 on each branch is associated with a simple decision tree that models the changes produced when the basic action is executed and the condition is true. Each simple decision tree is two levels deep, with a single chance node as the root, and a number of branches stemming from this node and ending with leaf nodes; each leaf node contains variable assignments specifying what happens after taking the action. For example, figure 1.b shows the Boolean tree of the basic action **IPG†; the** simple decision tree associated with the first branch states that when IPG is performed and thigh DVT is present ***(thigh-dvt = 1),*** then with a chance of 0.95 the test result is positive ***(IPGResult := +)*** and ***cost*** is increased by $120 ***(cost := cost*** + 1201; with a chance of 0.05 the test result is negative ***(IPGResult :=*** – ! and ***cost*** is increased by $120. The Boolean tree of **RUS** is identical to that of **IPG** but the test has higher specificity and higher cost.

We then construct Boolean trees that describe the composite actions using the Boolean trees of the basic actions. In the DVT domain, composite actions are used to describe situations where a certain basic action is performed only when certain conditions are true. For example, we might want to perform

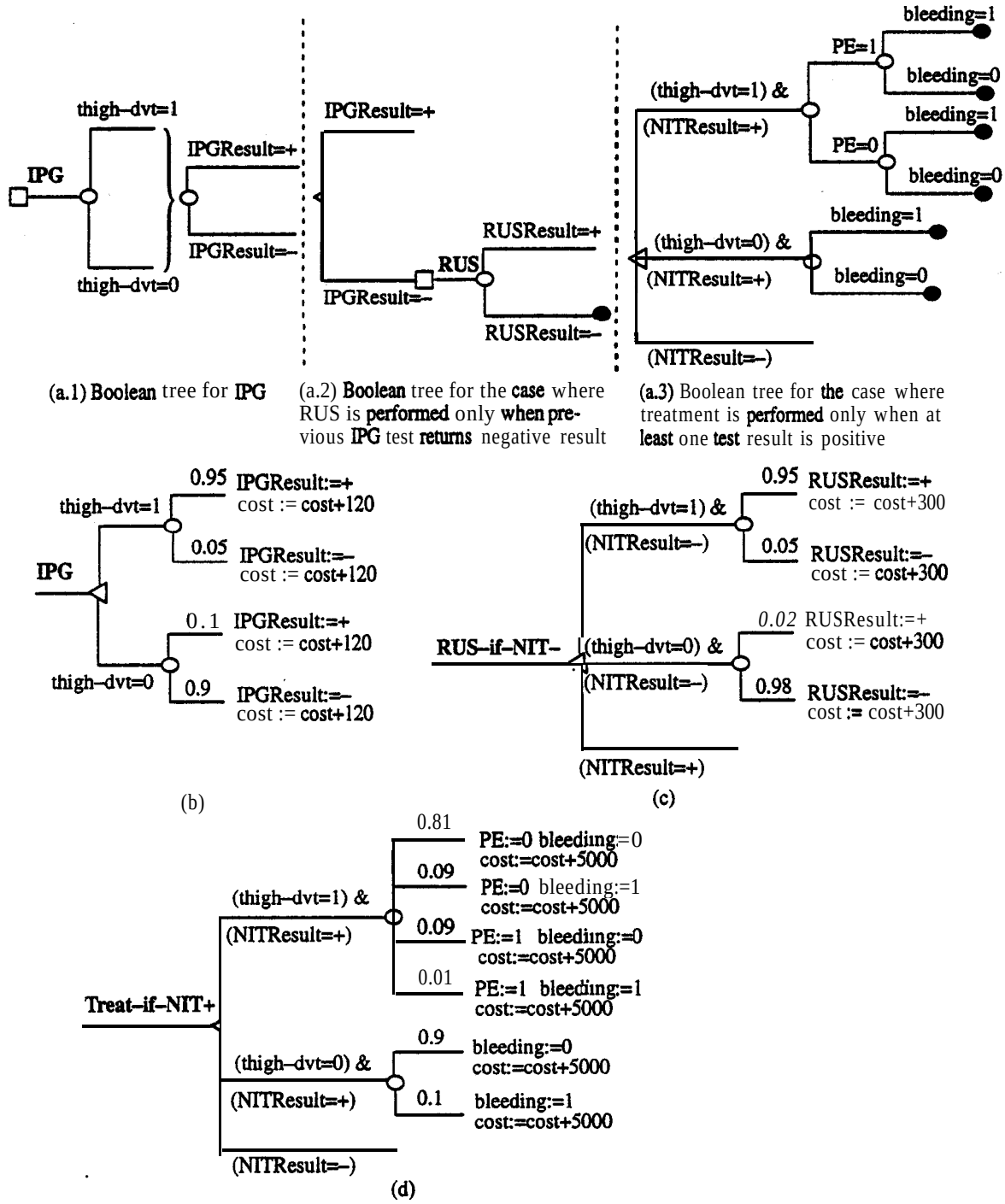

FIGURE 1. a, a decision tree represented as a sequence of three Boolean trees. The Boolean tree shown in part (a.1) is also a subtree. NIT is shorthand for "noninvasive tests" and an expression such as NITResult = + is shorthand for the expression (IPGResult = +) or (*RUSResult* = +). The symbol ◁ is used to indicate the switch of a Boolean node. The symbol } indicates all branches to which a subtree must be appended in order to create the complete decision tree. b-d, samples of action descriptions. The notation "cost := cost + *x*" indicates that the variable cost is incremented by *x* dollars.

the action **RUS** only when *IPGResult* is negative. We can represent this with the composite action **RUS if NIT-** shown in figure lc. Notice the similarity between this Boolean tree and the Boolean tree in figure la.2. Figure ld shows the Boolean tree for **Treat if NIT+,** which represents the case where patients are treated if at least one test result is positive. Notice also the similarity between this Boolean tree and the Boolean tree of figure la.3. For the rest of the paper, the Boolean tree of a basic or composite action will be called its *action description.*

Now we can represent a strategy with a sequence

of **(basic or** composite1 actions. For example, the strategy represented with the Boolean trees in figures la.l-a.3 can be represented by the sequence of actions **IPG, RUS if NIT-, Treat if NIT+.** We refer to a sequence of actions representing a strategy as a ***plan. Evaluating*** a plan, i.e., computing the plan's outcomes, is the process of constructing the complete decision tree corresponding to the plan and using the constructed tree to compute the outcomes.

In summary, we model the set of possible actions among which the decision maker can choose with a set of action descriptions, specifying the effects of each action. Each strategy being considered is represented with a sequence of action descriptions and is called a plan. Computing the outcomes of a strategy amounts to using the descriptions of the actions in the plan representing the strategy to construct a decision tree and using the constructed tree to compute the outcomes.

Notice that the method we have outlined for creating the basic and composite action descriptions encourages modularity of description. The action descriptions are created outside the context of their use in any particular decision tree and are then composed in sequence to create descriptions of decision trees representing the various possible strategies.

The DRIPS Planner

In the DRIPS planner, a decision-theoretic planning problem is described in terms of a set of action descriptions, using the STRIPS-style probabilistic action model, a set of prior probabilities, and one or more utility functions or outcome attributes. The goal of DRIPS is to find the optimal plan or set of plans.

When using a single utility function, DRIPS identifies the plan that maximizes expected utility. When using multiple utility functions or **outcome** attributes, DRIPS identifies the set of plans undominated with respect to the functions or attributes. In multiattribute utility theory this set is called the efficient frontier.[12] In all but the simplest of domains, analysis of each possible plan by exhaustive enumeration would be computationally prohibitive. **DRIPS** provides a method for abstracting probabilistic actions; it can reason with these abstractions in such a way that suboptimal classes of plans can be eliminated without explicitly examining all plans in those classes.

Action Abstraction

The DRIPS planner uses a technique that is called ***inter-action abstraction***[13] to search efficiently through the space of possible plans. Inter-action abstraction groups together a set of analogous actions and characterizes the set by the features common to all the actions. We then can plan with the abstract action and infer properties of a plan involving any of its instances. Formally, an inter-action abstraction of a set of actions $\{a^1, a^2, \ldots a^n\}$ is an action that represents the disjunction of the actions in the set. The actions in the set are called the ***instantiations*** of the abstract action and are considered to be alternative ways of realizing the abstract action.

To create an inter-action abstraction of a set of action descriptions $\{a^1, a^2, \ldots a^n\}$ we group the branches (or paths) of the action descriptions into disjoint sets such that each set contains at most one branch from each action description. For each set s that contains fewer than n branches, add $n - |s|$ branches, each with the effect of one of the branches already in the set and with condition False and probability zero. The effect of an abstract branch is any sentence entailed by each of the effects of the branches in the set. The condition is the disjunction of the conditions on the branches in the set. The probability is specified as a range: the minimum of the probabilities of the branches in the set and the maximum of the probabilities of the branches in the set.

In figure 2 the action **NIT** lnoninvasive test) is an abstraction of the actions **IPG** and **RUS.** The description of **NIT** is produced by grouping topologically corresponding branches. For example, the third branch of **IPG** is grouped with the third branch of **RUS.** The effect of the resulting abstract branch can be any expression that encompasses the effects of the abstracted branches. The two tests have the same effect on ***NITResult*** but different costs, so the cost of the abstract effect is represented by a range that includes the two costs. The two branches also differ in the probabilities of the effects, so we must specify with what probability the abstract effect is realized. If **IPG** is chosen as the instantiation, it is realized with probability 0.90, and if **RUS** is chosen, it is realized with probability 0.98. Thus, the effect is realized with a probability in the range 10.90 0.981. For these two actions the conditions on the grouped branches are identical, so the abstract branch is labeled with that condition. In general, grouped branches need not have the same condition.

The construction of abstract actions takes some effort on the part of the user. Rather than specifying tight abstractions of actions, the user is free to specify trivial abstractions. But the presence of informative abstractions, such as **NIT,** greatly improves the efficiency of the planner. We have developed tools to facilitate the specification of basic actions and the construction of abstract actions. An action-descrip-

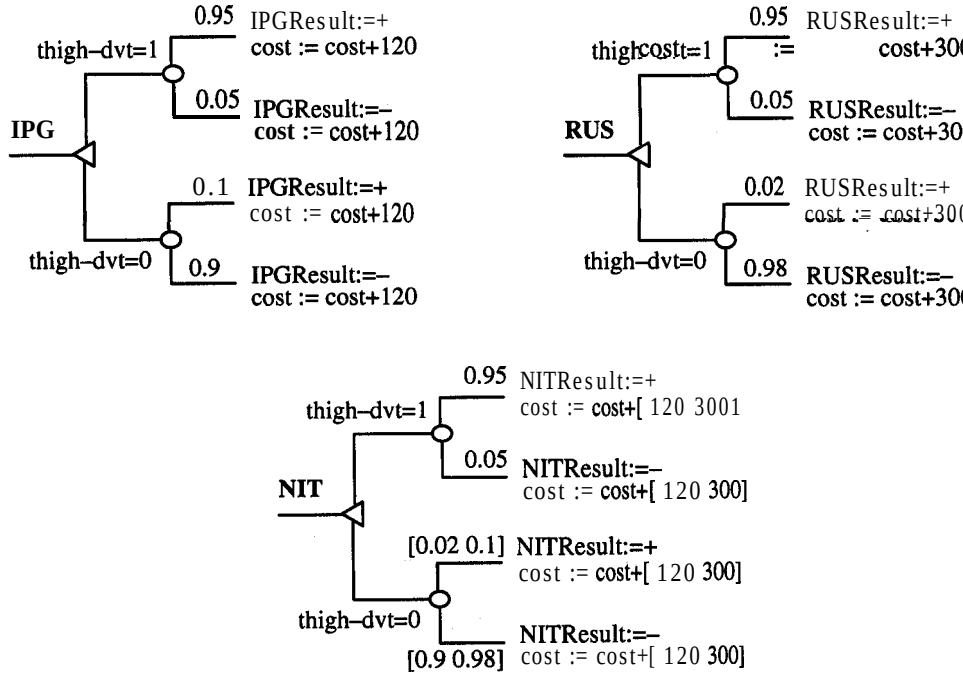

FIGURE 2. Abstraction of the basic actions IPG (a) and RUS (b) into noninvasive test NIT (c).

tion editor provides action templates that are easily filled in by the user. It provides syntax checking of the completed action descriptions and some consistency checking, e.g., to ensure that probabilities sum to one. Two abstraction tools partially automate the process of creating abstract actions.[14] An inter-action abstraction tool generates abstract action descriptions given a specification of which actions to combine and how to group their branches. A composition tool generates a decomposable action given a sequence of actions to compose.[15] The abstraction tools virtually eliminate the possibility of introducing errors in the process of constructing abstractions, since abstracting actions in the ways permitted by the tools is always correct.

The Abstraction/Decomposition Network

DRIPS searches through a space of plans structured into an ***abstraction/decomposition network.*** *An* abstract action has one or more sub-actions, which themselves may be abstractions or basic actions. A decomposable action has one or more sub-plans that all must be executed in sequence. The network for the DVT domain is shown in figure 3.

The most abstract action, **Manage DVT,** is an abstraction of six actions: **No Tests and Treat, Veno Tests, NIT Tests, Two Tests, Three Tests, Four Tests.** The number of tests represents the length of the longest allowed sequence of tests. Each of these actions further decomposes into a sequence of actions. For example, **NIT Tests** decomposes into **NIT, Treat NIT.** We assume, as in the original model, that no testing will be performed after the decision to treat is made. **NIT** can be instantiated as **IPG** or **RUS.** Our model for management of suspected DVT encompassed 6,206 concrete plans; for example, one complete plan (an instance of the **Two Tests** action) is **"IPG, Wait 7d if NIT-, Veno if NIT-, Treat if NIT + or Veno Thigh+,"** which means perform IPG, if the IPG is negative wait seven days and then perform venography, treat if IPG or venography shows thigh DVT.

The DRIPS Algorithm

DRIPS finds optimal plans by building abstract plans, comparing them, and refining only those that might yield optimal plans. Unlike "generative" planning schemes that build plans by adding actions,[10,11] DRIPS performs "refinement" planning. It begins with a set of abstract plans, and subsequently refines the plans from more general to more specific. Since evaluating abstract plans results in inferring probability intervals and attribute ranges, evaluating plans with respect to a single utility function assigns an expected utility interval to the abstract plan, which includes the expected utilities of all possible instances of that abstract plan. An abstract plan can be eliminated if the upper bound of its expected utility interval is lower than the lower bound of the expected utility interval for another plan. In the case of multiple utility functions, this property must apply

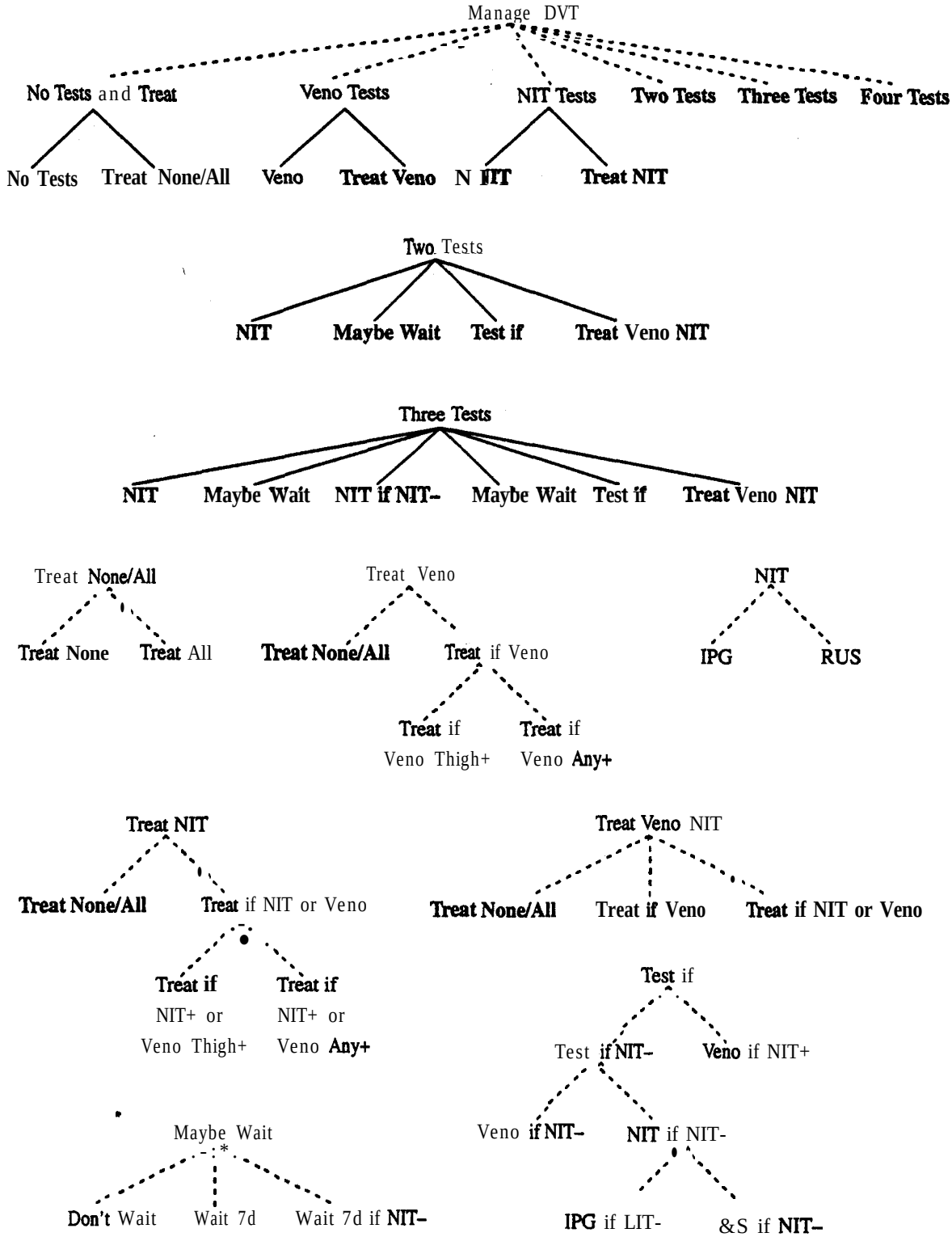

FIGURE 3. Abstraction/decomposition network. Abstraction relations are show-n with dashed lines and decomposition relations are shown with solid lines. Actions shown in bold have decompositions or abstractions that are displayed elsewhere in the figure. Due to space limitations, the subnetwork for the **Four Tests** sequence is not shown.

to all pairs of utility intervals. Eliminating an abstract plan amounts to eliminating all its possible instances. When the planner has some abstract plans with overlapping expected utility intervals, it refines the plans by instantiating some of their actions. This causes the expected utility intervals to narrow. If some of the new intervals do not overlap, more plans can be eliminated and the planner can then

refine those remaining one step further.

Given an abstraction/decomposition network, DRIPS evaluates plans at the abstract level, eliminates suboptimal plans, and refines remaining candidate plans further until only optimal plans remain. The algorithm works as follows.

1. Create a plan consisting of the single top-level action and put it into the set **plans.**

2. Until there is no abstract plan left in **plans,**
   - Choose an abstract plan **P.** Refine P by replacing an abstract action in P with all its instantiations, yielding a set of subplans $\{P_1, P_2, \ldots, P_n\}$. For each instantiation that has a decomposition, replace it with the sequence of actions into which it can be decomposed.
   - Compute the expected utility of all the subplans.
   - Remove **P** from **plans** and add $\{P_1, P_2, \ldots, P_n\}$.
   - Eliminate suboptimal plans in **plans.**

3. Return **plans** as the set of optimal plans.

Since **DRIPS** eliminates only plans that it can prove are suboptimal and if run to completion it explores the entire space of possible plans, it is guaranteed to find the optimal plans.

Table 1 • Prior and Conditional Probability Estimates*

Thigh DVT	
Prior probability	0.30
Progression to PE if untreated	0.50
Progression to PE if treated	0.10
Calf DVT	
Prior probability	0.10
Progression to thigh DVT in 7 days **if untreated**	0.20
Progression to thigh DVT in 14 days if untreated	0.25
Progression to thigh DVT if treated	0
Pulmonary embolism (PE)	
Proportion occurring within 7 days of identified thigh DVT	0.50
Proportion occurring within 14 days of identified thigh DVT	0.60
Fatality rate	0.10
Anticoagulation	
Bleeding episode	0.10
Proportion of bleeding episodes that are major	0.50
Proportion of major bleeding events that are fatal	0.05
Venography	
Venography-induced thigh DVT	0.01
Fatal contrast reaction	0.0001

***DVT** = deep **venous thrombosis; PE = pulmonary embolism.**

Table 2 • Characteristics of Tests Used in the Model*

	Sensitivity	Specificity
Thigh DVT		
Venography	0.96	0.96
Impedance plethysmography (IPG)	0.95	0.90
Real-time ultrasonography (BUS)	0.95	0.96
Calf DVT		
Venography	0.90	0.95

*DVT = deep **venous thrombosis.**

Table 3 • Cost Estimates for Tests, Treatment, and Complications

Tests	
Venography	$450
Impedance plethysmography (IPG)	120
Real-time ultrasonography (RUS)	300
Treatment	
Anticoagulation (7 days of intravenous hepa1 rin, 3 months of warfarin)	5,000
Complications	
Minor hemorrhage	1,500
Major hemorrhage	5,000
Nonfatal pulmonary embolism	10,000
Death (due to pulmonary embolism, hemorrhage, or contrast reaction)	30,000

Analysis of the DVT Domain

We encoded the DVT domain using data from Hillner et al.“ The prior probabilities, as well as the probabilistic effects of all the basic actions, are shown in tables 1 and 2. The abstraction/decomposition network representing a space of 6,206 possible strategies is shown in figure 3.

Rather than use a single utility function, we evaluated plans according to two factors: the number of deaths per 1,000 patients and the average cost per patient, as in the reference study. The cost was defined as the sum of the costs of tests and treatment and the costs associated with the state of the patient at the end of the plan. The cost values used are shown in table 3.

Given this problem description, DRIPS enumerated the efficient frontier by eliminating all plans that were dominated by some other plan along both evaluation factors. For the baseline case where cost of fatality was set at $30,000, the probability of thigh DVT at 0.3 and the probability of calf DVT at **0.1,** DRIPS enumerated the efficient frontier consisting of five plans after evaluating less than 1,000 plans, yielding a pruning rate of 83%. The plans in the efficient frontier are shown in table 4, listed in order

Table 4 • Best Clinical Strategies in Baseline Analysis*

Strategy	Deaths per 1,000 Patients	Average Cost per Patient, $	Additional Cost per Each Additional Life Saved, $
No test, treat none	15.9	1,905	—
Perform IPG, don't wait, perform venography if IPG was positive, treat if venography showed thigh DVT	5.47	2,359	43,700
Perform IPG, don't wait, perform venography if IPG was positive, treat if venography showed thigh or calf DVT	5.36	2,400	473,000
Perform RUS, treat if RUS showed thigh DVT	5.20	2,444	243,000
Perform RUS, don't wait, perform venography if RUS was negative, treat if RUS showed thigh DVT or venography showed thigh or calf DVT	4.50	3,276	1,610,000

*DVT = deep venous thrombosis; IPG = impedance plethysmography; RUS = real-time ultrasonography.

of decreasing total deaths per 1,000 patients.

The results of our analysis differed significantly from those of Hillner et al. First, our analysis determined a lower average cost per patient than the cost shown for all strategies in the reference study. Analysis demonstrated that instead of showing calf DVT progressing to PE through thigh DVT, as defined in their model, the reference study implicitly modeled calf DVT leading directly to PE with probability 0.5.[16] Second, two of the strategies considered to be among the best clinical strategies by the reference study did not appear in our efficient frontier. Both of these strategies involved waiting between tests. Third, our efficient frontier contained five strategies, while the reference study lists only three best strategies. For example, the reference study did not consider such cost-effective strategies as "perform IPG, don't wait, perform venography if IPG was negative, treat if venography showed thigh DVT." This can be accounted for by the fact that the DRIPS system explored more than 6,000 strategies, while the reference study examined only 24 strategies considered to be of clinical interest. Thus it may be advantageous to use an automated planner to sift through a large set of possible strategies rather than to use clinical judgment to select a small subset to analyze. The DRIPS planner provides the computational efficiency to make this feasible.

Analysis of Computational Efficiency

Since one of the primary goals in developing the DRIPS planner was to exploit the technique of abstraction to gain computational efficiency, it is important to quantitatively evaluate the effectiveness of

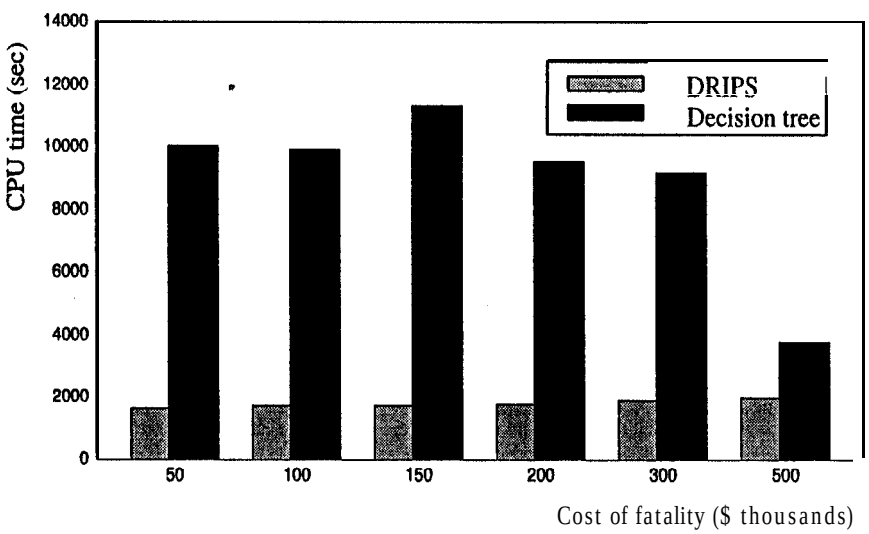

FIGURE 4. **Running times for DRIPS and a branch-and-bound decision-tree-evaluation algorithm for various costs of fatality. Running times are measured in CPU seconds on a DEC 5000/240.**

the approach. Theoretical analysis shows that with an appropriate abstraction hierarchy, the run-time complexity of DRIPS is exponentially better than that of exhaustive enumeration.[3] While this indicates great ***potential*** benefit, it does not provide a good indication of how effective the system will be in practice nor of how it compares with standard approaches. To provide such a measure, we used both DRIPS and a branch-and-bound decision-tree-evaluation algorithm to find the strategy with the lowest average cost per patient over a range of cost functions. We built software that used the basic action descriptions to generate a decision tree representing all possible plans and evaluated it using branch-and-bound technique8 to find the optimal plan. Both DRIPS and the decision-tree algorithm were implemented in the CommonLisp programming language, and analyses were run on the same machine.

Figure 4 shows the running times for DRIPS and for the decision-tree branch-and-bound algorithm at values of cost of fatality ranging from $50,000 to $500,000. DRIPS outperformed the branch-and-bound algorithm at all values. In the most extreme case, the running time of DRIPS was only 15% that of the branch-and-bound algorithm. While the running time for DRIPS varied little as the cost of fatality was changed, the running time for the branch-and-bound algorithm was highly sensitive to the cost of fatality and decreased considerably at a cost of fatality of $500,000. This is due to the fact that at that cost, plans that have a significant probability of fatality look sufficiently bad to be pruned away early.

To examine how the efficiencies of the two approaches vary as a function of problem size, we applied each approach to four versions of the DVT domain of increasing size. The first contained only plans including at most one test, for a total of **158** plans; the second contained plans with zero, one, or two tests, for a total of 1,022 plans; the third domain was the one mentioned above; and the fourth domain contained plans with up to five tests, for a total of 37,310 plans. Figure 5 shows the running times per plan for DRIPS and the branch-and-bound algorithm for each of the domains. The running time per plan for the branch-and-bound algorithm increases markedly (from 0.86 second per plan to 1.96 seconds per plan) as a function of problem size, while the running time per plan for DRIPS actually decreases (from 0.32 second per plan to 0.21 second per plan). This means that the overall running time for DRIPS increases more slowly than the domain size increases and the overall running time for the branch-and-bound algorithm increases more rapidly. Thus, DRIPS becomes even more efficient than the branch-and-bound algorithm as the size of the domain increases. The memory usage of DRIPS also compares favorably with that of the branch-and-bound algorithm over this same suite of problems (fig. 6). In the most extreme case, DRIPS uses only 4.4% as much memory as the branch-and-bound algorithm.

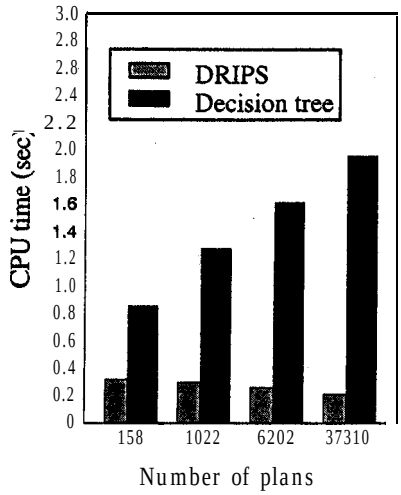

FIGURE 5. Running times per plan for DRIPS and a branch-and-bound decision-tree-evaluation algorithm for problems of increasing size. Times were obtained on a DEC 5000/240 with cost of fatality set at $50,000.

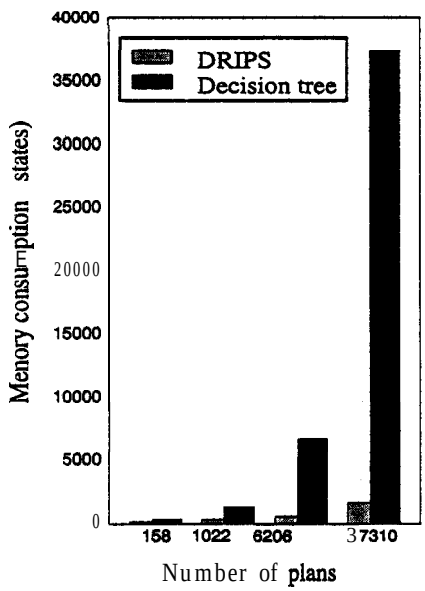

FIGURE 6. Memory consumption for DRIPS and a branch-and-bound decision-tree-evaluation algorithm for problems of increasing size. Values represent the maximum numbers of world states that need to be stored at one time by each algorithm. Values were obtained on a DEC 5000/240 with cost of fatality set at $50,000.

Discussion

Decision-theoretic refinement planning, as embodied in the DRIPS planning system, may be very useful for a wide variety of medical problems. The

current investigation has shown its applicability to problems of policy development, but the planner also could be applied to the management of individual patients. Strengths of this approach include the modularity of the action representation, and the ability to efficiently explore a large set of possible strategies. While the results of the empirical comparison of the performance of DRIPS against that of the branch-and-bound algorithm are not conclusive, they suggest that DRIPS can provide better performance in domains where good abstractions are available. Conclusive results await further empirical testing on a variety of problems.

Wellman's SUDO-PLANNER system[18] uses decision-theoretic principles to eliminate classes of suboptimal plans in domains characterized by partially satisfiable goals and actions with uncertain effects. It eliminates only those classes of plans that it can prove are dominated without resorting to reasoning about tradeoffs. Planning knowledge is represented in the form of qualitative probabilistic networks, which encode qualitative constraints on the joint probability distribution over a set of variables, as well as the qualitative relations of those variables to a value node. SUDO-PLANNER does not produce plans, but rather provides a set of constraints that the optimal plan must satisfy. Other recent work on decision-theoretic planning has included the probabilistic least-commitment planner of Kushmerick et al.[11] and several Markov-process-based planners.[19-22]

Uckun's SPIN system[23] uses abstraction/decomposition networks for protocol-based treatment planning, plan execution, and execution monitoring. The hierarchical representation of plans is used for controlling execution. Although the hierarchy encodes alternative plans, Uckun does not show how choices are made when alternatives exist. The SPIN system does not include decision-theoretic measures of uncertainty or plan quality.

The use of DRIPS offers several practical advantages. DRIPS allows the model builder to provide "modules" of information: the description of each action can be highly compartmentalized. One can easily alter a model's underlying assumptions to evaluate the effects on the outcome; in this way, DRIPS can perform sensitivity analysis of the model. Additional testing and treatment actions can be added easily to DRIPS planning models. Furthermore, applying DRIPS to a decision problem does not require the user to limit analysis to a small set of possible strategies. We view decision-theoretic refinement planning as a technique that has wide applicability to medical decision making.

The efficiency of our approach depends largely on how domain regularities are exploited to build the abstraction hierarchy. The applicability and the performance of the planner would be improved if methods could be devised to automatically construct effective abstraction hierarchies that minimize information loss due to abstraction. Devising such procedures has been shown to be possible for a limited class of domains.[21] In more complex domains with more expressive utility functions, it is much harder to define procedures to automatically generate good abstraction hierarchies. We are currently working on this problem.

Other issues we are addressing include improving system performance and expanding the class of problems the system can address. The current DRIPS software is written in the Common Lisp programming language, and due to its developmental nature, has not been optimized for run-time efficiency. The DRIPS system's algorithms are well suited for parallel processing, and can be adapted easily to run efficiently on a local- or wide-area network of general-purpose workstation computers. The DRIPS algorithm can currently solve planning problems involving discrete decisions, but many decision problems involve choosing levels of continuous quantities as well, e.g., the quantity of a drug to administer. We are extending the action representation and algorithm to handle such continuous decision variables.

The authors thank Maureen Good Finigan for helping run the analyses of the domain model. The DRIPS software is available via www at ⟨http://www.cs.uwm.edu/faculty/haddawy⟩.

References

1. Tversky A, Kahneman D. Judgement under uncertainty: heuristics and biases. Science. 1974;185:1124-31.
2. Kahneman D, Slovic P, Tversky A (eds). Judgement under Uncertainty: Heuristics and Biases. Cambridge, UK: Cambridge University Press, 1982.
3. Haddawy P, Suwandi M. Decision-theoretic refinement planning using inheritance abstraction. Proc Second Int Conf on AI Planning Systems, Chicago, IL, June 1994, pp 266-71.
4. Hillner BE, Philbrick JT, Becker DM. Optimal management of suspected lower-extremity deep vein thrombosis: an evaluation with cost assessment of 24 management strategies. Arch Intern Med. 1992;152:165-75.
5. Landefeld S, McGuire E, Cohen AM. Clinical findings associated with acute proximal deep vein thrombosis: a basis for qualifying clinical judgment. Am J Med. 1990;88:382-8.
6. Weinmann EE, Salzman EW. Deep-vein thrombosis. N Engl J Med. 1994;331:1630-41.
7. Pauker SG, Kassirer JP. Decision analysis. N Engl J Med. 1987;316:250-8.
8. Lau J, Kassirer JP, Pauker SG. Decision maker 3.0. Med Decis Making. 1983;3:39-43.
9. Hanks S. Projecting Plans for Uncertain Worlds. PhD thesis, Yale University, New Haven, CT, January 1990.
10. Goldman RP, Boddy MS. Epsilon-safe planning. Proc Tenth Conference on Uncertainty in Artificial Intelligence, Seattle, WA, July 1994, pp 253-61.
11. Kushmerick N, Hanks S, Weld D. An algorithm for probabilistic least-commitment planning. Proc Twelfth Nat Conf on

Artificial Intelligence. Seattle, WA, American Association for Artificial Intelligence, 1994, pp 1073-B.
12. Keeney RL, Flaiffa H. Decisions with Multiple Objectives: Preferences and Value Tradeoffs. New York: Wiley, 1976.
13. Haddawy P, Doan A. Abstracting probabilistic actions. Proc Tenth Conf on Uncertainty in Artificial Intelligence, Seattle, WA, July 1994, pp 270-7.
14. Finigan MG. Knowledge acquisition for decision-theoretic planning. Proc MAICSS'95, Carbondale, IL, April 1995, pp 98-102.
15. Doan A, Haddawy P. Generating macro operators for decision-theoretic planning. Working Notes of the AAAI Spring Symposium on Extending Theories of Action, Stanford, CA, American Association for Artificial Intelligence, March 1995, pp 68-73.
16. Kahn CE Jr, Haddawy P. Management of suspected lower-extremity deep venous thrombosis (letter). Arch Intern Med. 1995;155:426.
17. Lawler EL, Wood DE. Branch-and-bound methods: a survey. Oper Res. 1966;14:699-719.
18. Wellman MP. Formulation of Tradeoffs in Planning under Uncertainty. London, UK: Pltman, 1990.
19. Drummond M, Bresina J. Anytime synthetic projection: maximizing the probability of goal satisfaction. Proc Eighth Nat Conf on Artificial Intelligence, Boston, MA, American Association for Artificial Intelligence, July 1990, pp 138-44.
20. Dean T, Kaelbling LP, Kit-man J, Nicholson A. Planning with deadlines in stochastic domains. Proc Eleventh Nat Conf on Artificial Intelligence, Washington, DC, July 1993, pp 574-9.
21. Boutiller C, Dearden R;. Using abstractions for decision-theoretic planning with time constraints. Proc Twelfth Nat Conf on Artificial Intelligence, Seattle, WA, American Association for Artificial Intelligence, July 1994, pp 1016-22.
22. Cassandra AR, Kaelbling LP, Littman ML. Acting optimally in partially observable stochastic domains. Proc Twelfth Nat Conf on Artificial Intelligence, Seattle, WA, American Association for Artificial Intelligence, July 1994, pp 1023-B.
23. Uckun S. Instantiating and monitoring treatment protocols. Proc Eighteenth Annu Symp on Computer Applications ln Medical Care (SCAMC94), Washington, DC, November 1994, pp 689-93.