In: Proceedings of the 19th Annual Symposium on Computer Applications in Medical Care (SCAMC95), 1995.

Decision-Theoretic Refinement Planning: A New Method for Clinical Decision Analysis*

AnHai Doan, B.S., Peter Haddawy, Ph.D., Charles E. Kahn, Jr.,M.D.
Department of Electrical Engineering and Computer Science,
University of Wisconsin - Milwaukee, Milwaukee, Wisconsin

Clinical decision analysis seeks to identify the optimal management strategy by modelling the uncertainty and risks entailed in the diagnosis, natural history, and treatment of a particular problem or disorder. Decision trees are the most frequently used model in clinical decision analysis, but can be tedious to construct, cumbersome to use, and computationally prohibitive, especially with large, complex decision problems. We present a new method for clinical decision analysis that combines the techniques of decision theory and artificial intelligence. Our model uses a modular representation of knowledge that simplifies model building and enables more fully automated decision making. Moreover, the model exploits problem structures to yield better computational efficiency. As an example we apply our techniques to the problem of management of acute deep venous thrombosis.

INTRODUCTION

Clinical decision analysis, an analytic approach to decision making based on decision theory, is increasingly used to seek optimal decisions (or "strategies") for clinical protocols as well as management of individual patients [1].

Typically, a decision analyst will construct a set of decision trees to evaluate potential management strategies. There are two disadvantages to decision trees. First, even with the help of currently available software packages, constructing and evaluating decision trees remains tedious, and requires meticulous attention to avoid introducing errors. Second, the algorithms to identify the optimal strategy in large, clinically relevant decision problems may be computationally infeasible. These problems must be addressed if decision analysis is to gain widespread practical use and to be integrated into large decision support systems.

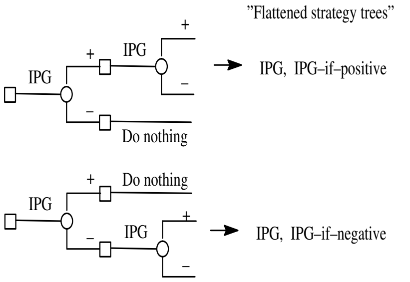
The area of artificial intelligence known as decision-theoretic planning has yielded new, alternative approaches for decision-making models. We introduce the abstraction-based model [2], which addresses the problems identified earlier in the decision tree model. We applied this abstraction-based model to a clinical problem to demonstrate the model's practical advantages over the decision trees. We refer to the full paper [3] for more detailed description of the model, and discussion of additional issues.

ACUTE DEEP VENOUS THROMBOSIS

Appropriate management of patients with suspected acute deep venous thrombosis (DVT) of the lower extremities remains an important and complex clinical problem. The clinical findings of DVT do not permit diagnosis with certainty [4]. Unchecked, DVT of the calf veins (*calf-DVT*) can progress to the deep veins of the thigh (*thigh-DVT*), which can cause pulmonary embolism (*PE*), a condition that entails significant morbidity and mortality. Anticoagulation therapy for DVT is expensive and carries the risk of hemorrhage. Even diagnostic procedures such as venography entail risks such as iatrogenic thrombosis or contrast reaction.

We constructed a model for diagnosis and treatment of DVT based on data from an article that compared 24 different management strategies [5]. The available diagnostic tests are venography (Venography), impedance plethysmography (IPG), and real-time ultrasonography (RUS). The physician can perform a combination of tests before deciding whether to treat; the decision of whether to perform a subsequent test, or treat, or do nothing at all, is based on the findings of previous tests. Tests can be performed in immediate succession, or can be separated by a waiting interval. So the physician can perform nine basic actions: Venography, IPG, RUS, don't test, treat, don't treat,

*This work was supported in part by the National Science Foundation grant #IRI-9207262 (PH), the 1993 American Roentgen Ray Society Scholarship (CEK), and National Library of Medicine grant LM05705 (for Integrated Advanced Information Management Systems [IAIMS] planning at the Medical College of Wisconsin).



(a)

```
(add–action  IPG

  (cond ((and  (thigh–dvt = 1)
               (dead = 0) (pe = 0))
         (1   (NITResult = +)    0.95
              (NITResult = –)    0.05
              (cost = cost + 120)  1))

        (and (thigh–dvt = 0)
             (dead = 0) (pe = 0))
         (1   (NITResult = –)    0.9
              (NITResult = +)    0.1
              (cost = cost + 120)  1))

        (or (dead = 1) (pe = 1))
         (1  ))))
```

(b)

```
(add–action  IPG–if–negative

  (cond ((and  (thigh–dvt = 1) (NITResult = –)
               (dead = 0) (pe = 0))
         (1   (NITResult = +)    0.95
              (NITResult = –)    0.05
              (cost = cost + 120)  1))

        (and (thigh–dvt = 0) (NITResult = –)
             (dead = 0) (pe = 0))
         (1   (NITResult = –)    0.9
              (NITResult = +)    0.1
              (cost = cost + 120)  1))

        (or (NITResult ≠ –)
            (dead = 1) (pe = 1))
         (1  ))))
```

(c)

Figure 1: (a) Mapping between strategy trees and plans (b) Action descriptor as specified in the DRIPS planner for IPG (c) Action descriptor for the composite action IPG_if_negative

wait, don't wait, and do nothing. Each basic action describes what is true after performing the action, conditioned on what is true before the action. The descriptions of these actions are supplied by the physician. Figure 1.b shows for example how the action of performing the IPG test is described in our model using Lisp code. The first line specifies the action's name as IPG. The second through the sixth lines describe the case where *thigh-DVT* is present (thigh-dvt = 1), the patient is not dead (dead = 0) and does not have diagnosed pulmonary embolism (pe = 0). Here NIT stands for "noninvasive tests", and NITResult refers to the result of a noninvasive test (IPG or RUS). In the above case performing IPG will give a positive result with probability 0.95 ((NITResult = +) 0.95), which means that a negative result will occur with probability 0.05 ((NITResult = -) 0.05); the cost of the test is $120 and is indicated by saying the overall cost is increased by $120 (cost = cost + 120). The next five lines describe the effects for the case where *thigh-dvt* is not present. The last two lines specify that in the case the patient is dead or has diagnosed pulmonary embolism the test is not performed.

THE ABSTRACTION-BASED DECISION MODEL

We begin with some terminology for the decision tree model. The user supplies a set of *basic actions*, e.g., the set of nine actions described earlier for the DVT domain. The decision problem has many *stages*; in each stage we choose to perform a basic action based on the findings of the actions performed in the earlier stages. For example, if in the first stage we choose to perform IPG, then in the second stage we can decide to perform a basic action (a second IPG, say) or not based on whether the outcome of IPG is positive or negative; four decision combinations can thus be formulated for the second stage, and we call such a combination a *stage policy*.

Assume now that in the second stage we are considering two policies: either perform IPG if the first IPG's outcome is positive, and do nothing if negative; or perform IPG if the first IPG's outcome is negative, and do nothing if positive. This will create in the tree model two subtrees representing two incomplete strategies (Figure 1.a).

In our model each strategy can be represented by "flattening" its corresponding tree, and we now proceed to show how. Consider the two strategies mentioned above. We first create two *composite actions* representing two considered policies at the second stage: IPG_if_negative has the effect of IPG on the branches where the first test's outcome is negative, and the effect of doing nothing on the other branches; IPG_if_positive has the effect of IPG on the branches where the first test's outcome is positive, and the effect of doing nothing on the other branches. Figure 1.c shows the description of action IPG_if_negative, which is created from the description of IPG.

We now can "flatten" two subtrees representing two strategies into two *plans*, each is a sequence of (basic or composite) actions appearing in the stages of the tree. For example the first strategy becomes the plan IPG, IPG_if_negative,$\cdots$ (Figure 1.a). Calculating the expected utility of this plan means first restoring the tree representing the strategy, then using the tree to calculate the expected utility for the root node, which is returned as the plan's or strategy's expected utility. In this manner, for all strategies represented in a (set of) decision tree(s), each can be represented with a plan. To do this, all considered stage policies must first be converted into composite actions. We note that the number of stage policies worthy of consideration tends to be small; they can be identified without ever building the decision tree, and the creation of composite actions involves very little work from the user.

So far we have only shown an alternative representation to the decision tree model. The strength

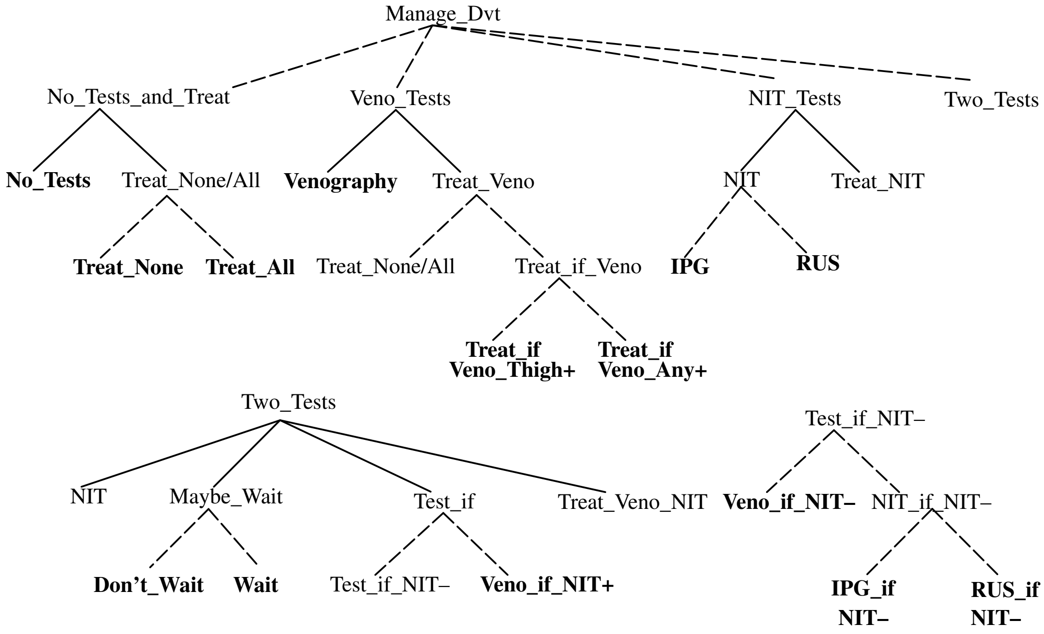


Figure 2: A partial description of the abstraction hierarchy (tree) for the DVT domain. Inter-abstractions are shown with dashed lines and sequential abstractions are shown with solid lines. Actions in bold letter are leaves, denoting composite or basic actions.

of this new representation is the ability to perform abstraction to reduce computation. Consider the above two plans. Since in the first stage we can also choose to perform RUS instead of IPG, we can add RUS to the first stage to have four plans instead of two. Since IPG and RUS are very similar, it seems reasonable to try to abstract these two tests into a new test NIT (noninvasive test). We now have only two plans: p_1 as NIT, IPG_if_negative, $\cdots$; and p_2 as NIT, IPG_if_positive, $\cdots$. We say IPG and RUS are two *instantiations* of the action NIT; plan IPG, IPG_if_negative, $\cdots$ is a *subplan* of plan p_1, obtained by replacing the first action in plan p_1 with one of its instantiations. An action created by asbstracting a set of actions is called an *abstract action.* A plan containing only basic or composite action(s) is called a *concrete plan*, otherwise it is an *abstract plan.* We require the abstraction technique to ensure that for each abstract plan a numeric interval called the *expected utility* of the plan can be computed in a manner similar to computing the expected utility of a concrete plan, and the interval of the expected utility of a plan includes the expected utility of any of its subplans. We have reported abstraction techniques satisfying these requirements in [6].

Consider the case when the lower bound of plan p_1's utility interval is greater than the upper bound of plan p_2's utility interval, from the abstraction requirements it follows that any subplan of p_2 has lower expected utility than any subplan of p_1, and therefore cannot be the optimal plan. We can eliminate the plan p_2, obtain two subplans of p_1, calculate their expected utilities, and choose the one with higher utility as the optimal plan (strategy). Note that the expected utilities of the subplans of p_2 are never calculated. The computational savings of our model rest on this observation. When eliminating a plan, the more subplans that plan contains, the higher the savings. We call abstracting a set of actions in a stage into a new action *inter-abstraction*. It is also possible to abstract actions in neighboring stages into an action, e.g., if before the second testing we insert a stage of performing the wait action, we can abstract the NIT action of the first stage with the wait action of the next stage, thus reducing the number of stages. This type of abstraction is called *sequential abstraction.*

We are now in a position to formalize our approach. The user first supplies a set of basic actions. He/she then proceeds to create composite actions according to stage policies deemed worthy of investigation. Next an abstraction hierarchy is specified which encodes all plans (strategies) worth consideration. A part of the abstraction hierarchy for the DVT domain in the case where only up to two diagnostic tests are allowed is shown on Figure 2. The hierarchy is actually a tree with composite or basic actions as leaves and abstract actions as its higher level nodes. The root Manage_DVT, the most abstract action, is an abstraction of four actions: No_Tests_and_Treat, Veno_Tests, NIT_Tests, Two_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 (sequentially abstracted from actions in the sequence). For example, NIT_Tests decomposes into NIT, Treat_NIT. Due to space limitations, the subtrees for the actions Treat_NIT and Treat_Veno_NIT are not shown. This hierarchy encompasses 1022 concrete plans; for example, one plan (an instance of the Two_Tests action) is "IPG, Wait, Veno_if_NIT-, Treat_if_Veno_Any+" (Treat_if_Veno_Any+ is obtained from refining Treat_Veno_NIT). The user now feeds the descriptions of the composite actions, the hierarchy, and a function specifying how the utility of an outcome is computed (e.g., specifying utility to be direct health care cost, the probability of survival, or a combination of both). The planner (computer) performs the following algorithm to locate the optimal plan.

1. Create a plan consisting of the root 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 $\{P_1, P_2, ..., P_n\}$. For each instantiation that is sequentially abstracted, replace it with the sequence of actions from which it is abstracted.
   - Compute the expected utility of all the subplans.
   - Remove P from plans and add $\{P_1, P_2, ..., P_n\}$.
   - Eliminate suboptimal plans in plans by comparing their utility intervals.
3. Return plans as the set of optimal plans.

This algorithm has been implemented as the DRIPS (decision-theoretic refinement planning) system. Since DRIPS only eliminates plans that it can prove are suboptimal and if run to completion explores the entire space of possible plans, it is guaranteed to find the optimal plan(s).

RESULTS

To investigate the applicability of our approach to the area of medical decision making we used the DRIPS system to evaluate the DVT domain described earlier, adding the case of performing up to three tests. This new DVT domain encompasses 6,206 plans. For symplicity in this paper we define the optimal strategy to be the one with the lowest direct health care cost. Based on the "standard

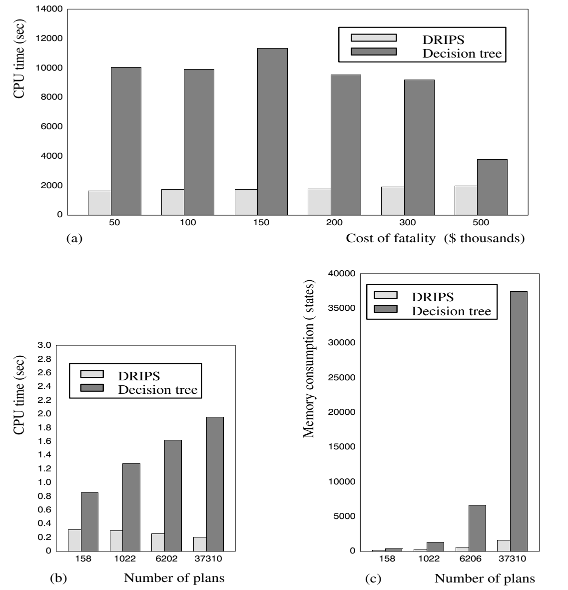


Figure 3: (a) Run times for DRIPS and a branch-and-bound decision tree evaluation algorithm for various costs of fatality (b) Run times per plan and (c) memory comsumption for DRIPS and the branch-and-bound algorithm for problems of increasing size. Memory consumption values represent the maximum memory size that is required at one time by each algorithm.

model" assumptions of the original model [5], in which the cost of a fatal event was set at $30,000, DRIPS determined that the plan with the highest expected utility was their Strategy A, "No Tests, No Treatment." The cost of this plan differed from the cost identified in the reference manuscript. In fact, for several other plans, DRIPS calculated cost values that differed from those published in the source manuscript. In attempting to resolve this discrepancy, we discovered that at least one of the 24 decision trees constructed by Hillner and colleagues was incorrect [7].

Evaluating this model on our DEC 5000/200 machine with unoptimized code written in Common Lisp took less than eight minutes. This good performance is due to the fact that DRIPS used the abstraction hierarchy to discard 5,551 (89%) of the 6,206 possible management strategies without explicitly identifying them, and calculating their expected utilities. Such pruning is important to gain computational efficiency in large models.

We also used DRIPS to perform a simple one-way sensitivity analysis on the cost of a fatal event. The optimal plan remained "No tests, no treatment." for a cost of fatality between 0 and $73,630, At values for cost of fatality between $73,630 and $200,000, the optimal plan became "Perform IPG, don't wait, perform venography if IPG was positive (Veno_if_NIT+), and treat only if venography shows thigh DVT (Treat_if_Veno_Thigh+)."

In order to demonstrate the computational efficiency of DRIPS, we compared it to a standard branch-and-bound algorithm for evaluating decision trees. The comparison was done by running both algorithms on the DVT domain, varying domain size and parameters. Figure 3.a shows the running time for DRIPS and the decision tree branch-and-bound algorithm on the DVT domain of up to three tests and cost of fatality ranging from \$50,000 to \$500,000. DRIPS outperforms the branch-and-bound algorithm at all values. We also applied both algorithms to four versions of the DVT domain in which the maximal number of tests allowed is varied from one to four. Figure 3.b shows the run times per plan for DRIPS and the branch-and-bound algorithm for each of the domains. The run time per plan for the branch-and-bound algorithm increases markedly (from .86 seconds per plan to 1.96 seconds per plan) as the domain size increases while the run time per plan for DRIPS actually decreases (from .32 seconds per plan to .21 seconds per plan). This means that the relative efficiency of DRIPS over the branch-and-bound algorithm increases as the domain size increases. Figure 3.c shows that the memory usage of DRIPS also compares favorably to that of the branch-and-bound algorithm over this same suite of problems. In the most extreme case, DRIPS uses only 4.4% as much memory as the branch-and-bound algorithm.

DISCUSSION

We have demonstrated empirically that our approach has far better computational resource utilization than that of the decision tree model. As we argued, this is necessary if clinical decision analysis is to be successfully integrated into decision support systems. The time efficiency of our algorithm also allows the user to relax symplifying assumptions made on the model and to consider more strategies, thus permitting more comprehensive examination of the domain.

Although creating the composite actions and the abstraction hierarchy is easy, calculating the descriptions for abstract actions based on the composite actions proved to be a tedious job. We are working on automating this calculation process [8]. We are also investigating how to automatically specify the optimal abstraction hierarchy. Our approach works with any abstraction hierarchy, but some are better than the others since they provide better pruning. This is currently a topic of intensive research in planning; the main idea is to exploit domain structure and utility function regularities to decide what actions are best abstracted together.

Our approach also allows the model builder to provide "modules" of information: the descriptions of each action can be highly compartmentalized. Action modifications are performed by the user only on the small set of basic and composite actions, thus avoiding potential construction and modification errors associated with building decision trees.

References

[1] Kassirer JP Paulker SG. Decision analysis. *New Engl J Med*, 316:250–258, 1987.

[2] P. Haddawy, A. Doan, and R. Goodwin. Efficient decision-theoretic planning: Techniques and empirical analysis. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Montreal, August 1995. To appear.

[3] P. Haddawy, A. Doan, and C.E. Kahn. Decision-theoretic refinement planning in medical decision making: Management of acute deep venous thrombosis. Submitted to Journal of Medical Decision Making. Available via www at http://www.cs.uwm.edu/faculty/haddawy.

[4] Landefeld S, McGuire E, Cohen AM. Clinical findings associated with acute proximal deep vein thrombosis: a basis for qualifying clinical judgment. *Am J Med*, 88:382–388, 1990.

[5] 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*, 152:165–175, 1992.

[6] A.H. Doan and P. Haddawy. Decision-theoretic refinement planning: Principles and application. Technical Report TR-95-01-01, Dept. of Elect. Eng. & Computer Science, University of Wisconsin-Milwaukee, January 1995. Available via anonymous FTP from `pub/tech_reports` at `ftp.cs.uwm.edu`.

[7] CE Kahn, Jr and P Haddawy. Management of suspected lower-extremity deep venous thrombosis (letter). *Archives of Internal Medicine*, 155:426, February 1995.

[8] A. Doan and P. Haddawy. Generating macro operators for decision-theoretic planning. In *Working Notes of the AAAI Spring Symposium on Extending Theories of Action*, Stanford, March 1995.