A Comparison of Action-Based Hierarchies and Decision Trees for Real-Time Performance*

David Ash **Barbara Hayes-Roth**

KSL / Stanford University
701 Welch Rd. Bldg. C
Palo Alto, CA 94304-1709
ash@sumex-aim.stanford.edu

Abstract

Decision trees have provided a classical mechanism for progressively narrowing down a search from a large group of possibilities to a single alternative. The structuring of a decision tree is based on a heuristic that maximizes the value of the information gained at each level in the hierarchy. Decision trees are effective when an agent needs to reach the goal of complete diagnosis as quickly as possible and cannot accept a partial solution. We present an alternative to the decision tree heuristic which is useful when partial solutions do have value and when limited resources may require an agent to accept a partial solution. Our heuristic maximizes the improvement in the value of the partial solution gained at each level in the hierarchy; we term the resulting structure an *action-based hierarchy*. We present the results of a set of experiments designed to compare these two heuristics for hierarchy structuring. Finally, we describe some preliminary work we have done in applying these ideas to a medical domain--surgical intensive care unit (SICU) patient monitoring.

0 Introduction

Traditionally, decision trees have provided a mechanism for progressively narrowing down a search of a wide range of possibilities to a single alternative efficiently. However, in this context efficient has generally meant that the agent needs to reach its goal after consuming the minimal amount of resources. Suppose, however, that its goal is to diagnose one of a number of faults, and that the agent does not have the resources to reach its goal before it needs to act to remedy the situation. If the domain is such that partial information about the correct alternative is much better than nothing, then we will define the effectiveness of our solution not merely by how efficiently the agent reaches its goal but also by the relative worth of the partial answers it finds along the way. In this paper, we present an alternative to the traditional decision tree, *the action-based hierarchy*, which takes into account the value of partial information and performs better when this is factored in.

Our ideas build on several themes in the literature. The basic structure of the action-based hierarchy we will describe is quite similar to that of decision trees as used by Quinlan [QU83]. Other researchers ([FA92], [CH91]) have addressed the question of how to structure a decision tree, as we do, but without specifically concerning themselves with the value of the partial answers reached along the way. We follow the reactive planning approach taken in [AG87], [NI89], [RO89], and [SC87]. The algorithm which is utilized with an action-based hierarchy at run-time may be viewed as a type of *anytime algorithm* [DE88]. Our ideas are also connected to the work on bounded rationality ([DE88], [HO87], [RU91]); we guarantee response without consuming more than a certain maximal amount of expendable resources.

1 Action-Based Hierarchies

Action-based hierarchies are used to discriminate among a set of alternatives or *faults*. Each leaf node in the hierarchy corresponds to a single fault; higher nodes in the hierarchy correspond to classes of faults. The class of faults associated with a node is always the union of the classes of faults associated with its children, and the top node has associated with it the

* This research was supported by DARPA through NASA grant NAG2-581 (ARPA order 8607) and the DSSA project (DARPA contract DAAA21-92-C-0028). We acknowledge the many useful discussions we have had with our Guardian colleagues Adam Seiver, David Gaba, Juli Barr, Serdar Uckun, Rich Washington, Paul-Andre Tourtier, Vlad Dabija, Jane Chien and Alex Macalalad. Thanks to Ed Feigenbaum for sponsoring the work at the Knowledge Systems Laboratory.

set of all faults. In order to help the agent achieve its goal, there are *tests* available that it can perform. Each outcome of a test will identify the correct fault as being in one of a number of not necessarily disjoint subsets of the set of all faults. Each test has an associated *cost* which is a heuristic estimate of the difficulty of performing the test. For instance, in a real-time domain, the cost of a test might be the time consumed by the test.

There are also *actions* which can be performed; the agent tries to identify the correct fault because it needs to find an action to remedy that fault. Therefore, for every (action, fault) pair there is a *value* representing a heuristic estimate of the value of performing a particular action if a particular fault is present. This value ranges from -1 (undesirable action) through 0 (neutral) to +1 (ideal solution).

The main idea behind an action-based hierarchy is that associated with each node in the hierarchy there is a corresponding action which has the highest expected value given that the agent knows the fault falls within the set associated with this node, but has not yet discriminated among the node's children. This action can then be performed if the agent is required to act (e.g. it reaches a deadline) but has not yet completed its diagnosis. The action might be suboptimal (depending upon which fault actually turns out to be present), but is likely to be much better than doing nothing.

Figure 1 shows a sample action-based hierarchy. There is a set of four faults F1, F2, F3, and F4, and the top of the diagram shows a hierarchy with a test used to distinguish among the children of each node. The faults associated with each node, as well as the action with the best expected value, are shown in the boxes at each level in the hierarchy. The box at lower right shows how performing each test enables us to distinguish among faults; the box at lower left gives values for each (action, fault) pair.

2 Hierarchy Structuring

Given a set of faults, and the associated tests, costs of tests, actions, and values of actions, there are many hierarchies that could be constructed, although not all such hierarchies would be equally useful. Indeed, a hierarchy could be constructed by starting with the root node, and recursively selecting tests which help discriminate among the faults associated with the current node, and adding children to the current node with one child corresponding to each possible outcome of the test. More formally, the following algorithm could be run to structure a hierarchy for use by the agent in real time:

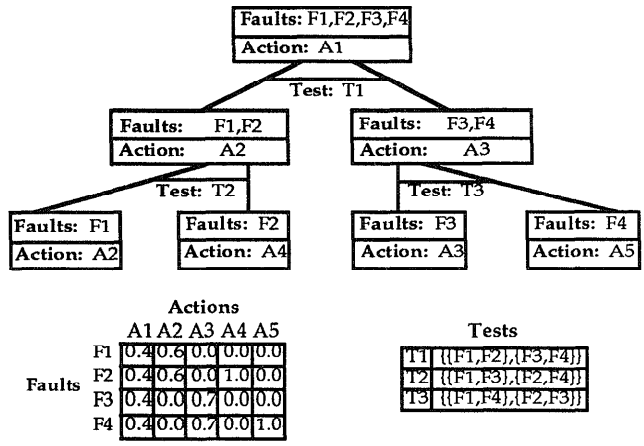

Figure 1: Sample Action-Based Hierarchy

❶ Start at the top of the yet-to-be-built hierarchy, associating the set of all faults with this top node.
❷ Pick a leaf node in the hierarchy in DFS order, or stop if there are no more leaf nodes to expand. Associate with this leaf node the action which has the highest expected value for the set of faults associated with this node.
❸ If the leaf node cannot be expanded further, go back to step 2 and pick another leaf node.
❹ Find all tests relevant to the set of faults associated with the current node.
❺ If no tests were found in step 4, go back to step 2 and pick another leaf node.
❻ Determine which test found in step 4 is best according to some heuristic.
❼ Expand the current node with one child corresponding to each possible outcome of the test found in step 6, and the associated fault sets suitably adjusted.
❽ Go back to step 2 and pick one of the children of the current node.

Note that this algorithm will never get stuck in a local maximum at any node as long as there are relevant tests for expanding a node. Once we have constructed a hierarchy in this manner, it is very simple to use it--the agent simply performs the following algorithm in real time:

❶ Label the root node of the hierarchy as the *current best hypothesis.*
❷ Perform the test associated with the current best hypothesis. If an action is required before this test can be completed, perform the action associated with the current best hypothesis.
❸ When the test results come back, refine the current best hypothesis to the one of its children pointed to by the result of the test.

2.1 Different Heuristics for Hierarchy Structuring

The variable we are most interested in in the current paper is the particular heuristic used in step 6 of the hierarchy structuring algorithm. The traditional decision tree approach is to take the *prior probability* $P(f)$ of each fault f, the set of faults F associated with the current node (a subset of all the faults), and the set of outcomes T of the test (each outcome is a subset of F), and then to compute the following function:

$$\sum_{t \in T} P(t)I(t) \text{ where}$$

$$P(t) = \sum_{f \in t} P_w(f) \ , \ I(t) = \sum_{f \in t} \frac{P_w(f)}{P(t)} \log \left(\frac{P_w(f)}{P(t)}\right)$$

$$\text{and } P_w(f) = \frac{P(f)}{\#\{t \in T : f \in t\}}$$

The effective result of this approach is that the agent maximizes the expected information content of the result of the test. This guarantees that the agent will reach its goal of diagnosing a single fault in the minimum possible time, but does nothing to guarantee that it will find actions of high value along the way. We take a different approach. In addition to the functions defined above, we use a cost function $C(t)$, and a value function $V(f,a)$ giving the value of action a for fault f. The function we wish to maximize is:

$$\frac{V(T)-V(F)}{C(T)} \text{ where } V(F_0) = \max_{a \in A} V(a,F_0)$$

and F_0 is a any set of faults. Also

$$V(T) = \frac{\sum_{t \in T} V(t)P(t)}{\sum_{t \in T} P(t)} \text{ and}$$

$$V(a,F_0) = \frac{W(a,F_0)}{P(F_0)} \qquad W(a,F_0) = \sum_{f \in F_0} V(f,a)P_w(f)$$

(Note that an assumption is being made here that if one fault appears in more than one of the outcomes of a test, then if the fault is present, each of those outcomes is equally likely. This assumption will be made throughout the paper.) The intuition above is that the agent performs the test which will yield the highest expected increase in the value of the best action available to date.

2.2 Evaluating Hierarchy Performance

In order to effectively compare the performance gained through the use of different hierarchies, we need to have a method of evaluating that performance. We assume that the cost of a test (the resources consumed by the test) is equal to the real time consumed by the test, but our ideas apply just as well to other types of cost which can be measured by a real number. Because of the assumed real time flavor, we will refer to the maximum time the agent is allowed to consume before acting as the *deadline.*

On any given occasion where the hierarchy is used and a fault is present, the agent will start at the top of the hierarchy and refine the current best hypothesis until it reaches a leaf node or its deadline. At any given point in time, therefore, there will be a current best hypothesis, an associated action, and a value of that action for the fault. It is this value which forms the basis for our evaluation. The value of the action associated with the current best hypothesis becomes a step function of time representing the performance of the hierarchy in one particular instance.

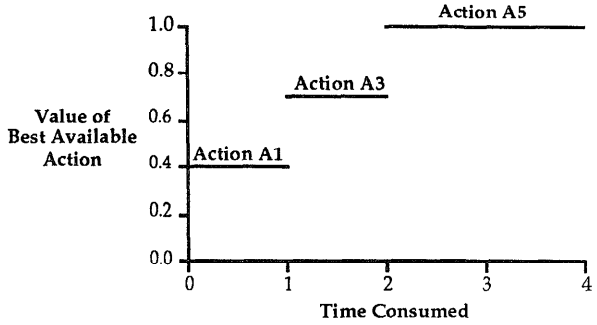

Figure 2: Hierarchy Performance on Fault F4

For example, Figure 2 shows the performance of the hierarchy illustrated in Figure 1 on the fault F4,

assuming that all tests have constant time (or cost) of 1. We can now define the average hierarchy performance as follows:

$$E(f,t) = \sum_{p_f} P(p_f)E(f,p_f,t)$$

where the summation is over all paths p_f from the root node to a leaf node corresponding to fault f, and $E(f,p_f,t)$ denotes the performance of the hierarchy over time assuming that the path p_f is taken to the leaf node. The function $P(p_f)$ denotes the probability that path p_f is taken given that fault f is present.

Having defined the performance of the hierarchy for a particular fault, we can define the overall performance of the hierarchy as the weighted average of the performances for each fault:

$$E(t) = \sum_{f \in F} P(f)E(f,t)$$

$E(t)$ gives the performance of the hierarchy at various possible deadlines t.

3 Formal Properties of Action-Based Hierarchies

It is possible to prove that under certain well-defined assumptions, the performance of action-based hierarchies is optimal. Specifically, we need to make the following assumptions:

❶ The values of tests are additive. The value of a test is given by the function $V(T)$ defined in section 2. Two tests T_1 and T_2 may be combined into a single test T by combining their results: $T = \{t_1 \cap t_2 : t_1 \in T_1 \wedge t_2 \in T_2\}$. Additivity means that the value of the combination is the sum of the values of each test: $V(T) = V(T_1) + V(T_2)$. In a similar manner, additivity must also apply to combinations of more that two tests.
❷ The value of any test or combination of tests is constant for all faults. That is, if this constant is 0.7 for a given combination of tests, then for *any* set of faults corresponding to particular outcomes of these test, the best action available will have value 0.7 for all the faults in the set.
❸ The deadline occurs immediately following a successful refinement of the current best hypothesis.
❹ The structuring of the hierarchy conforms with the algorithm given at the start of section 2.
❺ For any set of faults, there is always an action available which has positive expected value so that doing something is better than doing nothing.

The proof is based on the fact that the action-based hierarchy structuring algorithm is a greedy algorithm, and when test values are additive the agent does best by simply performing the apparently best test first without considering lookahead.

4 Formal Experiments

The assumptions outlined in the last section clearly do not hold in general. Therefore, we designed experiments to test the following two hypotheses:

❶ Using the action-based hierarchy will provide substantially better performance than the decision tree when evaluated as described in section 2.
❷ Using the decision tree will provide substantially better performance when only speed in reaching a leaf node matters.

The experimental method was to randomly assign prior probabilities, test outcomes, and values of actions for faults. The prior probabilities were assigned by dividing the interval [0,1] into uniformly distributed partitions. Each test was randomly assigned to divide the fault set into two equal parts (the number of faults was always even). No fault appeared as part of both outcomes of a particular test. The values of actions for faults were also randomly distributed in the interval [0,1].

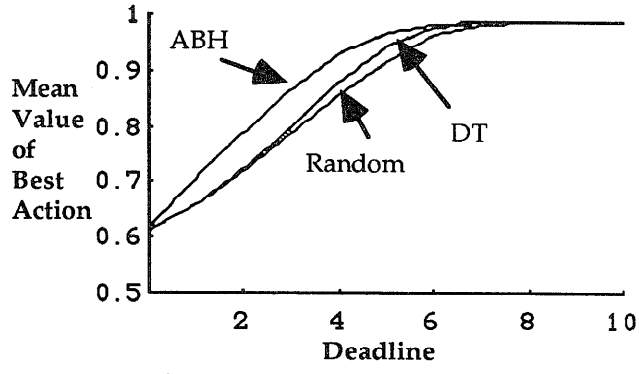

Figure 3: Hierarchy Performance with Different Deadlines and Heuristics

Figure 3 shows the difference in performance, over time, of action-based hierarchies, decision trees, and the randomly structured hierarchies. This shows

that action-based hierarchies always do at least as well as the other approaches, no matter what the value of the deadline (the amount of resources available for diagnosis), but that this advantage varies over time from nil at time zero, to a maximum for deadline values of around 3 or 4, back to nil for large deadline values. This corresponds, roughly speaking, to the fact that all hierarchies perform equally well at their root and their leaf nodes, but that at the intermediate nodes there is an advantage for certain hierarchies over others.

The reader may wonder why there are not certain time points at which decision trees would have an advantage over action-based hierarchies. The answer appears to be that for individual performance profiles DT might have an advantage at certain time points, but this charateristic is lost when we compute the average.

Now, suppose we are not interested in intermediate nodes but only in getting to a leaf node as quickly as possible. In Figure 4, we show a graph of the probability that a leaf node has been diagnosed by particular deadline values. Looking at this figure it is apparent that over a relatively small range of the deadline (between about 4 and 7) using a decision tree offers a very significant advantage over the other approaches. However, for other values of the deadline, it makes very little difference.

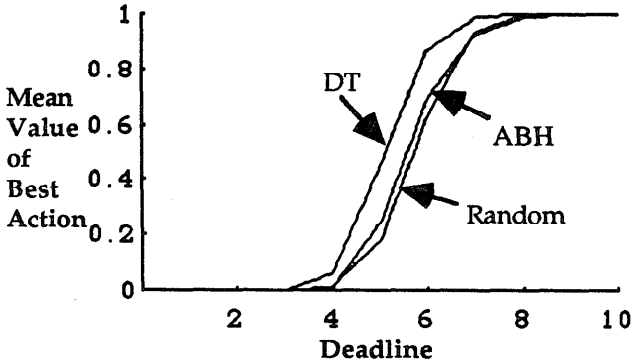

Figure 4: Hierarchy Performance using All-or-Nothing Evaluation

5 Implementation in a Medical Domain (ReAct)

We have used the idea of an action-based hierarchy to structure the knowledge base for a system, called ReAct, designed to provide fast response time in a surgical intensive care unit (SICU). ReAct is described in detail in [AS93] and is part of the Guardian SICU patient monitoring system [HA92]. Because it uses an action-based hierarchy, ReAct is able to meet deadlines. This is in contrast with other medical AI systems ([BR87], [CL89], [FA80], and [HO89]) which perform diagnosis efficiently but without specifically addressing deadline issues. Knowledge regarding faults, tests, costs of tests, probabilities of faults, etc., has been provided by a medical expert; the hierarchy structuring algorithm has been used to produce an action-based hierarchy. A small part of the hierarchy is shown in Figure 5. In applying these ideas to a real-world domain, we are interested in determining whether the same results as reported earlier in this paper apply. In this domain, the theoretical assumptions made in section 3 will definitely not apply; nor will the variables necessarily fit the random distributions used to run the experiments described in section 4.

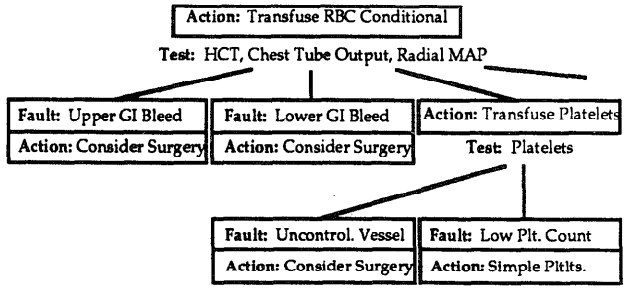

Figure 5: Portion of Medical Action-Based Hierarchy

Figure 6 shows the results of an experiment run using the medical problem similar to the ones described in section 4. In this experiment we assumed that all tests took constant time to come back (30 minutes). As can be seen, for small values of the deadline, action-based hierarchies enjoyed a modest advantage over decision trees. For larger values of the deadline (not shown here) there was no discernable difference. We anticipate that if we took test times into account, the advantage of ABH over DT would be much more noticeable, even for larger values of the deadline.

6 Future Work

In addition to the domain experiments described in section 5, we intend to explore a number of different avenues in our ongoing research. The first question is whether there are other, still better heuristics

which could be used to structure the hierarchy. We have tried combining the two heuristics, and we expect that a more sophisticated combining function than we have used could lead to behavior which is better than either algorithm in certain circumstances. Another question is whether better performance could be obtained by altering the basic structure of the solution. At present, only one test is performed at each node in the hierarchy. In the domain described in section 5, this approach would be the exception rather than the rule: generally physicians will order a whole battery of tests to help them distinguish among the children of the current node. Thus, our hierarchies should be able to perform similarly if we hope to achieve performance comparable to that obtained by physicians.

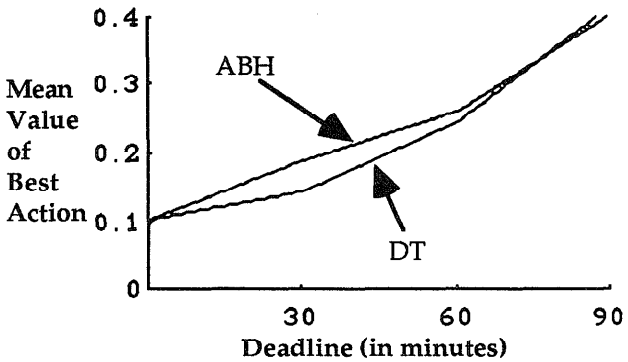

Figure 6: Hierarchy Performance in Medical Problem

Our measures for cost and value (numbers between 0 and 1) are quite crude and may not be sufficient in many domains. Therefore, we intend to explore the process of hierarchy structuring in domains where we have more sophisticated notions of cost and value.

References

[AG87] Agre, P., Chapman, D. Pengi: an implementation of a theory of activity. *Proceedings of the Sixth International Conference on Artificial Intelligence*, 268-272, 1987.

[AS93] Ash, D., et.al. Guaranteeing real-time response with limited resources. *Artificial Intelligence in Medicine*, 5(1):49-66, 1993.

[BR87] Bratko, I., Mozetic, I., Lavrac, N. Automatic synthesis and compression of cardiological knowledge. *Machine Intelligence 11*, pp 435-454, 1987.

[CH91] Chrisman, L., Simmons, R. Sensible planning: focusing perceptual attention. *Ninth National Conference on Artificial Intelligence*, 756-761, 1991.

[CL89] Clarke, J., et.al. TraumAID: a decision aid for managing trauma at various levels of resources. *Proceedings of the Thirteenth Annual Symposium on Computer Applications in Medical Care*, Washington, DC, November, 1989.

[DE88] Dean, T., Boddy, M. An analysis of time-dependent planning. *Seventh National Conference on Artificial Intelligence*, 49-54, 1988.

[FA80] Fagan, L. VM: representing time-dependent relations in a medical setting. PhD Thesis, Computer Science Department, Stanford University, June, 1980.

[FA92] Fayyad, U., Irani, K. The attribute selection problem in decision tree generation. *Tenth National Conference on Artificial Intelligence*, 104-110, 1992.

[HA92] Hayes-Roth, B., et.al. Guardian: a prototype intelligent agent for intensive-care monitoring. *Artificial Intelligence in Medicine*, 4:165-185, 1992.

[HO87] Horvitz, E. Reasoning about beliefs and actions under computational resource constraints. *Proceedings of the 1987 AAAI Workshop on Uncertainty in Artificial Intelligence*, 1987.

[HO89] Horvitz, E., et.al. Heuristic abstraction in the decision-theoretic Pathfinder system. *Proceedings of the Thirteenth Annual Symposium on Computer Applications in Medical Care*, Washington, DC, November, 1989.

[NI89] Nilsson, N.J. Action Networks. *Proceedings from the Rochester Planning Workshop: From Formal Systems to Practical Systems*, J. Tenenberg et al., ed., University of Rochester, 1989.

[QU83] Quinlan, J.R. Inductive inference as a tool for the construction of high-performance programs. In R.S. Michalski, T.M. Mitchell, and J. Carbonell, *Machine Learning*, Palo Alto, Calif.: Tioga, 1983.

[RO89] Rosenschein, S.J. Synthesizing information-tracking automata from environment descriptions. *Proceedings of Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, San Mateo, CA, 1989.

[RU91] Russell, S., Wefald, E. Principles of metareasoning. *Artificial Intelligence*, 49:361-395, 1991.

[SC87] Schoppers, M. Universal plans for reactive robots in unpredictable environments. *Tenth International Joint Conference on Artificial Intelligence*, 1987.