COGNITIVE SCIENCE 1, 158–192 (1977)

Decision Theory and Artificial Intelligence II: The Hungry Monkey*

JEROME A. FELDMAN
University of Rochester

ROBERT F. SPROULL
Xerox Palo Alto Research Center

This paper describes a problem-solving framework in which aspects of mathematical decision theory are incorporated into symbolic problem-solving techniques currently predominant in artificial intelligence. The utility function of decision theory is used to reveal tradeoffs among competing strategies for achieving various goals, taking into account such factors as reliability, the complexity of steps in the strategy, and the value of the goal. The utility function on strategies can therefore be used as a guide when searching for good strategies. It is also used to formulate solutions to the problems of how to acquire a world model, how much planning effort is worthwhile, and whether verification tests should be performed. These techniques are illustrated by application to the classic monkey and bananas problem.

1. INTRODUCTION

Mathematical decision theory is concerned with decision making under conditions of uncertainty. Because this is also a major concern of artificial intelligence, one would expect considerable interaction between the disciplines. Although there has been some (primarily in the study of search algorithms, see Hart, Nilsson, & Raphael, 1968), the two fields have presented basically competing paradigms. It is the purpose of this and related papers (Yakimovsky & Feldman, 1974) to show how artificial intelligence problems can be attacked by methods derived from decision theory. More generally, we are concerned with how to combine numeric and symbolic reasoning.

The central idea of mathematical decision theory is that a numerical utility function can be used to evaluate decisions (see Chernoff & Moses, 1959, or Raiffa, 1970, for introductions to decision theory). A single numerical value is used to summarize the advantages of a set of actions. A typical utility function would be the profits realized from a particular investment outcome. Although it

*This research was supported in part by the Advanced Research Projects Agency of the Department of Defense under contract DAHC 15-73-C-0435, ARPA order 2494, and in part by the Xerox Corporation. Reprints may be obtained from Jerome A. Feldman, Computer Science Department, University of Rochester, Rochester, N.Y.

might seem that the use of a single number could preclude the choice of an optimal strategy in some cases, the central theorem of decision theory shows essentially that this cannot happen. The presentation of this theorem is beyond the scope of this paper (see deGroot, 1970), for an elementary presentation), but the result can be stated simply: if various outcomes have known utilities and known probabilities of occurrence, then any acceptable (admissible) strategy is equivalent to one which maximizes expected utility. Different strategies arise from different assumptions about probability and utility functions. Both kinds of functions are subjective models of the behavior of the problem domain and may be quite complex.

The first, and perhaps the best, argument for a numerical utility function is that the choice between alternative courses of action is often intrinsically numerical. One chooses the cheapest, or fastest, or strongest alternative. We have found that many problems in robot problem solving are virtually inexpressible in nonnumerical terms.

Another main use of a numerical utility function is in "comparing the incomparable." If flying is faster and safer than driving, but more expensive and subject to delay, how can we choose which to do? What change in price would cause us to choose otherwise? The expected utilities of the alternative decisions answer these questions. By contrast, a heuristic program capable of comparing differing strategies must have rules covering all possible combinations of goals and circumstances, and the addition of new alternatives may require significant reprogramming. In a decision-theory formulation, much of the complexity in the tradeoff comparisons is embodied in the structure of the utility function. A vast controversy-filled literature testifies to the intricacies of utility and probability assessments that attempt to model tradeoffs made by humans (Tversky & Kahneman, 1974). Our objective is considerably simpler: the design of the utility function need address values only within a particular problem.

Our first major application of decision-theoretic methods in robotics attacked the problem of image segmentation (Yakimovsky & Feldman, 1974). The problem of segmentation, breaking a complex image into sections, is a central one in machine perception; the analogous problem arises in the analysis of speech (Woods, 1974) and in other complex problems. The approach applies Bayesian decision theory and problem-dependent information (semantics) to determine an acceptable segmentation of the image. This program was quite successful and the general ideas are being widely adopted (Garvey Tenenbaum, 1974; Tenenbaum, 1973).

However, the image segmentation effort did not explore all advantages of decision-theoretic formulations. Image analysis has been a peripheral problem in AI and has often been attacked by partially numerical techniques. We believe that many problems in AI can be clarified by abandoning a strictly "symbolic data processing" viewpoint and by employing decision-theoretic techniques.

The central problem addressed in this paper is that of planning and acting. This is a core issue in AI and becomes increasingly important as we begin to

apply AI techniques to real problems. The vehicle used for discussion in this paper is the classic "Monkey and Bananas" problem. It is a simple, well-known problem which can be extended naturally to include many of the basic issues we wish to address. These issues include planning under uncertainty, assessing costs and risks, error recovery, the value of information, and the cost of planning.

Because the monkey's world may seem rather artificial, the reader may prefer a more practical example. The issues of uncertainty, cost, risk, and information gathering seem easy to appreciate in the case of the "wheelless student," the real-life problem of buying a used automobile. A typical procedure is first to read newspaper advertisements and bulletin boards to assess the situation generally. Then, at relatively low cost, one can telephone various purveyors of cars and inquire about them. At some point, one must actually go to the effort of seeing and driving certain of these. There are professional diagnostic services which can be employed at considerable cost to further test the car. In each of these steps, one must decide when to stop that stage and go on to the next one. One does not, of course, proceed in strict order; there will normally be alternatives at several different levels of investigation.

Notice that the "plan" itself is trivial: read, telephone, look, drive, professionally test, and buy. It is the application of this plan to the world situation which is difficult. We believe that much intelligent activity is characterized by complex applications of simple plans, and this belief has led us to concentrate on the closely related questions of plan elaboration and execution.

The following four sections illustrate the applications of decision theory to AI with the monkey and bananas example. The first section describes the use of simple decision-theoretic techniques applied to symbolic problem solving. The subsequent sections illustrate a broader application of techniques, particularly for allocation of resources to planning and acting. The indented paragraphs describe generalizations of ideas in the example which may be ignored on first reading.

2. DECISION THEORY IN SYMBOLIC PROBLEM SOLVING

Decision theory helps a symbolic problem-solver search for the best plan that achieves a given goal. A utility function on plans can govern a search strategy that explores plans of high utility; the search terminates by announcing the plan of highest utility.

We shall illustrate how symbolic problem solving and decision analysis can be combined with the classical example: *A hungry monkey is in a room in which a bunch of bananas hangs from the ceiling. The monkey cannot reach the bananas. There is, however, a movable box in the room; if the box is under the bananas and the monkey stands on the box he can reach the bananas and eat.* The goal for a symbolic problem solver is to find a plan that will feed the monkey.

A typical problem solver is given a symbolic model of the problem and searches for a combination of "operators" that achieves a given goal. A possible symbolic model, specified in the style of a modern AI language, records informa-

tion about the position of objects with an "AT" assertion that associates an object and its Cartesian coordinate position. Additional assertions declare boxes to be climbable and pushable and bananas to be edible. An initial set of assertions might be:

(AT MONKEY 9 9 0)
(AT BANANAS 0 0 5)
(AT BOX 2 2 0)
(HEIGHT BOX 5)
(CLIMBABLE BOX)
(PUSHABLE BOX)
(EDIBLE BANANAS)

The operators are specified below. If all assertions in the list of preconditions are in the assertion data base, then the operator can be applied. Application of an operator causes assertions in the delete list to be deleted from the data base, and those in the add list to be added. (The functions X, Y, and Z refer to the coordinate entries in the AT predicate. The symbol \$ will match any value in the corresponding position in the assertion.)

WALKTO(α)
  Preconditions: (AT MONKEY \$ \$ 0)
  Delete list: (AT MONKEY \$ \$ 0)
  Add list: (AT MONKEY $X(\alpha)$ $Y(\alpha)$ 0)

PUSHTO(α,β)
  Preconditions: (AT MONKEY $X(\alpha)$ $Y(\alpha)$ 0)
  (PUSHABLE α)
  (AT α \$ \$ 0)
  Delete list: (AT MONKEY \$ \$ 0)
  (AT α \$ \$ 0)
  Add list: (AT MONKEY $X(\beta)$ $Y(\beta)$ 0)
  (AT α $X(\beta)$ $Y(\beta)$ 0)

CLIMB(α)
  Preconditions: (CLIMBABLE α)
  (AT MONKEY $X(\alpha)$ $Y(\alpha)$ 0)
  Delete list: (AT MONKEY \$ \$ 0)
  Add list: (AT MONKEY $X(\alpha)$ $Y(\alpha)$ HEIGHT(α))

CONSUME(α)
  Preconditions: (EDIBLE α)
  (AT MONKEY $X(\alpha)$ $Y(\alpha)$ $Z(\alpha)$)
  Delete list: (EDIBLE α)
  Add list: (FED)

The problem solver, given the goal (FED), would generate the plan:

WALKTO(BOX)
PUSHTO(BOX,BANANAS)
CLIMB(BOX)
CONSUME(BANANAS)

In a robotics experiment, the original assertions and this sequence of operators can be used as a set of commands to software and hardware subsystems that would cause a robot to simulate the actions of the monkey.

If the problem statement and the corresponding symbolic information given to the problem solver were expanded to include multiple tools, multiple sources of food, or multiple goals, the problem solver could generate other plans as well. If the initial assertions model several boxes, a plan for each box can be generated. However, if the symbolic model becomes at all large and intricate, the combinatorial explosion would overwhelm any present problem solver.

Computing the Utility of a Plan

The utility of any one of the WALKTO, PUSHTO, CLIMB, CONSUME plans is derived from a *utility model* that accompanies the symbolic model. It is a measure of the value of executing each of the steps in the plan and thus achieving the goal. We shall assume for now that this utility can be expressed as a sum of contributions from individual steps and a contribution representing the value of achieving the goal (*Note:* This eludes several important considerations such as risk which will be treated later.)

For each goal, we assign a function that evaluates the utility of achieving the goal. In our example, we shall assign $U_e = 200$ to the goal of eating, that is, a state in which the monkey is fed. Goals of less value to the monkey are assigned correspondingly smaller utilities. For later reference, we shall assume that the next most desirable goal is "don't bother trying to eat," which has utility U_{db}.

The utility associated with executing each step of the plan is often called the "cost" of the step. A robotics experiment that simulates each operator with a collection of processes, including computation and control of a robot vehicle or manipulator, might use cost assignments that express the consumption of resources required to accomplish each step. Table 1 specifies an assignment of negative-valued cost functions C that reflect the expenditure of resources required for each step.

WALKTO(α). The monkey walks from its present location (x_m,y_m) to $(X(\alpha),Y(\alpha))$. The cost is $C_w = -1 *$ distance$((x_m,y_m),(X(\alpha),Y(\alpha)))$.

PUSHTO(α,β). The monkey pushes the object α to the location $X(\beta),Y(\beta)$. $C_p = -10 *$ distance$((X(\alpha),Y(\alpha)),(X(\beta),Y(\beta)))$.

CLIMB(α). The monkey climbs the object α. $C_b = -20$.

CONSUME(α). The monkey consumes the food α. $C_c = -5$.

Using this model, the total utility of the symbolic plan WALKTO, PUSHTO, CLIMB, CONSUME is: $U_{total} = C_w + C_p + C_b + C_c + U_e$. The utility of the next best plan is U_{db}.

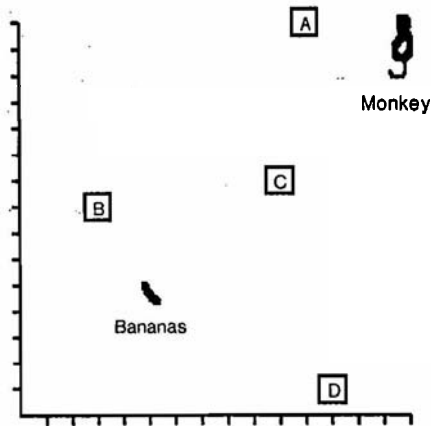

FIG. 1 Map showing the location of the monkey, the bananas and four boxes. The axes indicate an (x,y) coordinate system.

Comparing Alternative Plans

The plan with greatest utility can be selected for execution: it is the "best" of the plans generated by the symbolic problem solver, as evaluated by the utility model. To illustrate the power of comparing plans, consider generating plans using each of the four boxes shown in the "map" of Fig. 1. Table 1 shows the total utilities of the WALKTO, PUSHTO, CLIMB, CONSUME plans using the different boxes. The plan to use box B has the greatest utility and is therefore selected as the best plan. This simple utility model adds considerable capability to the problem solver. The location of the boxes and the cost functions determine which box is selected as the best one to use. For example, if $C_p = C_w$, box C will be preferred rather than B. If the initial position of the monkey changes, different boxes may be preferred. (Figure 2a shows a map of regions in which the monkey might start out, together with the preferred box in each region (the map is made assuming $C_p = -2$*distance).)

TABLE 1

	C_w	C_p	C_b	C_c	U_e	U_{total}
Box A	−3	−117	−20	−5	200	55
Box B	−13	−36	−20	−5	200	126
Box C	−6	−64	−20	−5	200	105
Box D	−13	−81	−20	−5	200	81

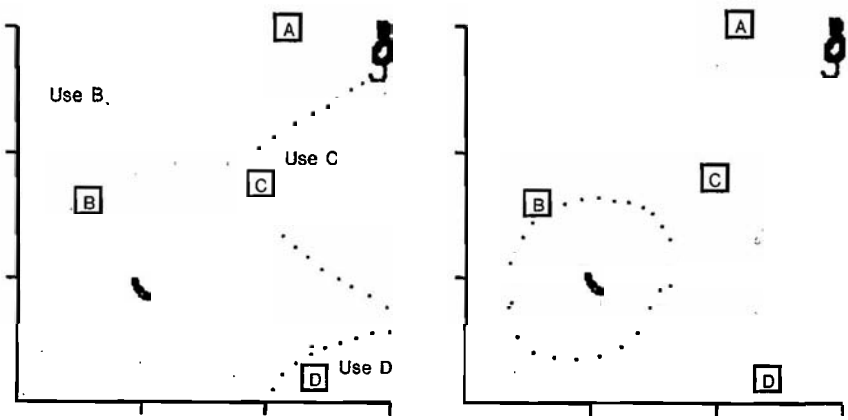

FIG. 2 (a) Regions of box preference by initial monkey position. (b) Region in which a box preferred to B must lie.

The cost functions also provide answers to a number of questions that an intelligent strategist must pose. For example, when should one try to find a box of higher utility than any presently located, and where should one search? (Figure 2b shows the region in which a box preferred to B would have to lie.)

Another important class of strategic questions concerns what decision theorists call "sensitivity analysis": how much confidence can be placed in the identification of the best plan? Is it substantially better than the next best, or do the utilities show that the planner is nearly indifferent to the choice? Do slight inaccuracies in the map or model cause a substantial change in the choice of best strategy? We shall later return to these important questions.

Coping with Uncertainty

Execution of a plan can go awry and produce outcomes considerably different from the desired goal. Clearly the reliability of a plan must be incorporated into the calculation of its utility. Decision theory shows how to weight the utility of an outcome with its probability of occurrence and thus to calculate a total utility that expresses the consequences of possible failures.

Let us augment the monkey-and-bananas problem by introducing a simple kind of failure: *There are two kinds of boxes in the room: wooden and cardboard. Cardboard boxes will not support the monkey; wooden ones will.*

When the plan outline is applied to a box of unknown type, either the box is wood and the monkey succeeds in eating, or it is cardboard and he fails (see Fig. 3). In the absence of more precise information about the box to which the plan is applied, we shall use a single probability p_0 to express the likelihood that the box is wooden. In addition, we shall assign a utility to the failure outcome. A simple assignment is U_{db}, corresponding to abandoning the quest for food. However,

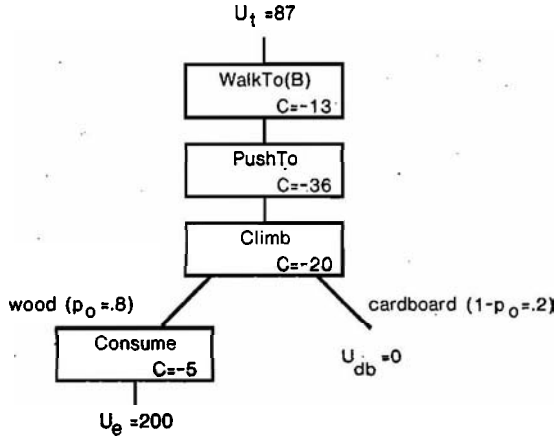

FIG. 3 A decision tree for a plan in which CLIMB may fail.

failure-recovery plans of higher utility may exist: a plan to clear away the destroyed cardboard box and to try using another box may have a higher utility than U_{db}. Techniques for devising failure recoveries will be more fully developed below; we shall temporarily assume the utility of the failure outcome to be U_{db}.

The utility of the plan is calculated as the mathematical *expectation* of the utilities of the individual outcomes, that is:

$$U_{total} = \Sigma\, p_i U_i$$

where U_i is the total utility of a particular path in the "decision tree," and p_i is the probability of taking the path ($\Sigma\, p_i = 1$). For Fig. 3, the total utility U_{total} is $p_0(C_w + C_p + C_b + C_c + U_e) + (1 - p_0)(C_w + C_p + C_b + U_{db})$.

This technique allows the planner to trade off cost and reliability; classical AI problem solvers have no means of expressing these tradeoffs. For example, if we use the costs of Table 1 and assume an identical p_0 for all boxes, no change occurs in the selection of the best plan. However, if the probabilities differ for various boxes, a reliable plan may be preferred to a less reliable one. For example, if p_{0c} is the probability that box C is wood, p_{0b} that of box B, and $p_{0c} > p_{0b} + .11$, the expected utility of using box C will be greater than that of using box B.

The expected utility is a numerical measure of the merits of the *strategy* expressed by the plan. It does not predict that executing the plan will have an outcome of comparable utility, but only predicts the *average* utility of outcomes of many executions. Thus, if we use the *expected utility* as a measure when searching for good plans, we do not guarantee good outcomes, only good strategies. (This notion is central to classical decision theory, see deGroot, 1970.)

Finding Good Plans

Because the utility of a plan can be used to compare the merits of competing plans, it can be used to guide a search for good plans. The basic idea is to search by expanding paths of greatest expected utility. A number of algorithms have been devised that can use numerical measures to guide such a search (see surveys in Lawler & Wood, 1966, and Nilsson, 1971). Thus, augmenting the symbolic model with a numerical utility model widens the range of applicable search techniques.

Using numerical measures to guide search is not new to AI. Many game-playing programs employ a numerical score to represent the desirability of a board position and to guide a search. In fact, a game-playing program that uses a plausible move generator and a numerical evaluation of progress toward a win is a simple example of a combination of symbolic and (ad hoc) utility models in problem solving. Robotics problem-solving programs (e.g., STRIPS, see Fikes & Nilsson, 1971)[1] have also used simple numerical measures, such as the number of operators in a plan, as a search guide.

Searching can be guided in several ways; we shall illustrate "progressive deepening" and "pruning" as examples. The A^* algorithm (Nilsson, 1971; Pohl, 1973) is typical of a progressive-deepening approach: a nonterminal node, N, of a search tree is expanded if it lies on the most promising path. The measure of promise is an estimate of the utility of the complete plan, computed as the sum of two terms: g, a measure of the "costs" ascribed to the nodes already included in the path (i.e., the total cost of the steps from the root node to N), and h, an upper bound estimate of the utility of a path from N to the goal. (Note: All "costs" are negative. Thus an upper bound on a set of costs is one for which resource expenditure is least.) These terms for the monkey and bananas example might be:

g(node) = $\Sigma\ C_i$ from root to node
h(node) = (upper-bound estimate of costs from node to the goal) + U_e

The calculation of g requires calculating the contribution to the utility of the steps of the partially complete plan. It is for this reason that we have formulated our utility model as a sum of terms attributable to individual steps of the plan. The estimate used for h can be based on a simple "state-difference" approach, for example, if, at node N, the monkey is not located at the bananas, then an upper bound on h is the cost of moving the monkey to the bananas.

Progressive deepening uses a running estimate of the path utility to guide application of further planning effort. One advantage of this technique is that it will automatically attenuate the processing of plans that loop: such plans are abandoned because, as steps are added to the plan, g decreases continually without an offsetting increase in h.

[1] Use of A^* was revealed in a private communication.

"Pruning" is characteristic of several kinds of search algorithms that avoid exploring portions of the tree because the optimal plan can be shown to lie elsewhere. Many algorithms in use in operations research, known generically as "branch and bound" algorithms, have this property. The basic idea is to ignore paths that have an upper bound on their utility that is less than the utility achievable by some other path. A similar technique, for minimax trees, is called "alpha–beta," which has been extended for use with decision trees (Nilsson, 1971; Pauker & Kassirer, 1975).

The key information that guides pruning is the *bounds* information: the tighter the bounds, the more pruning necessary. Bounding the utility of a plan such as that of Fig. 3 requires bounding the utility of the part of the plan that is incomplete, the failure path. Bounding the failure path is equivalent to bounding the utility of the possible recovery strategies. One way to do this is as follows:

Lower bound: Don't bother with the current goal, and assign utility U_{db} to the failure. Thus the lower bound is $U_l = C_w + C_p + C_b + p_0(C_c + U_e) + (1-p_0)U_{db}$.

Upper bound: Assume that the failure caused no damage, and that there is an alternative plan as good as the present one. (Note: Given that a good plan fails, we do not have to assume that there exists a *better* one, because that will be covered by cases involving other boxes.)

$$U_u = C_w + C_p + C_b + p_0(C_c + U_e) + (1-p_0)U_u$$
$$U_u = (C_w + C_p + C_b)/p_0 + C_c + U_e$$

If we perform these calculations for all boxes, as in Table 1, we obtain Table 2. This bounding scheme shows clearly that utilities of plans involving boxes A, C, or D cannot exceed even the lower bound on using box B. Thus portions of the tree that call for boxes A, C, and D to be used to reach the bananas are pruned.

The numerical utility model thus furnishes information that is useful in guiding search. This information, whether encoded in cost functions or in bounding schemes, can easily involve "domain-dependent" information, as exemplified by our assignment of a function of distance to the cost of walking. The symbolic model also constrains search: the symbolic preconditions are used to avoid searching foolish plans, for example, ones that reach for the bananas when the monkey is not nearby. This technique can be implemented in the new AI lan-

TABLE 2

Box	U_l	U_u
A	16	20
B	87	109
C	66	82
D	42	52

guages (see survey in Bobrow & Raphael, 1974) by instantiating each subgoal pursuit as a separate process and including bounds estimates and costs when proposing new subgoals. A branch-and-bound algorithm, such as *A**, can then schedule the processes (subgoals), always executing the most promising subgoal. Such dynamic allocation of effort to problem-solving processes motivated the design of the SAIL multiple process structure (Feldman, 1972).

3. IMPROVING THE PLAN

In this section, we will focus on improvements that can be made to a plan prior to its execution. A plan outline can often be altered to yield a greater expected utility by making detailed, often local, improvements to the plan. The measure of improvement in this *plan elaboration* process is the increase in expected utility resulting from filling in details.

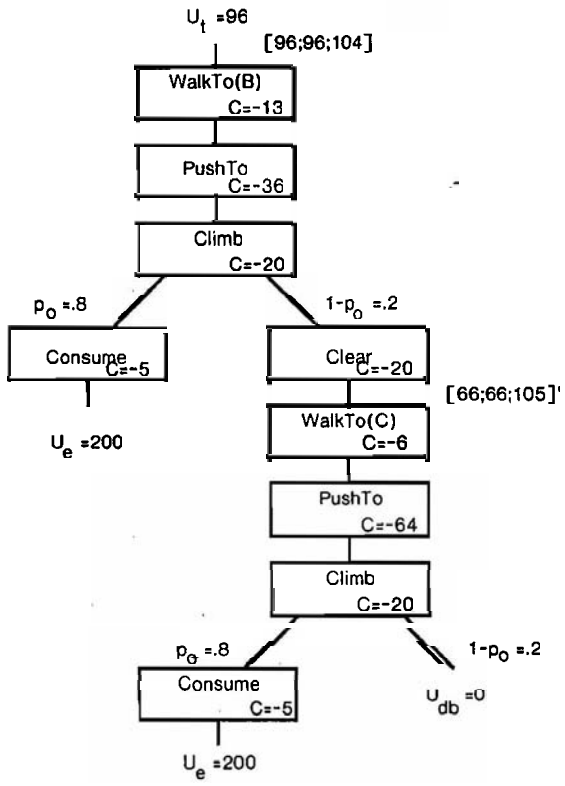

FIG. 4 Elaboration to increase the utility of failure in CLIMB. The numbers in brackets are explained in the text.

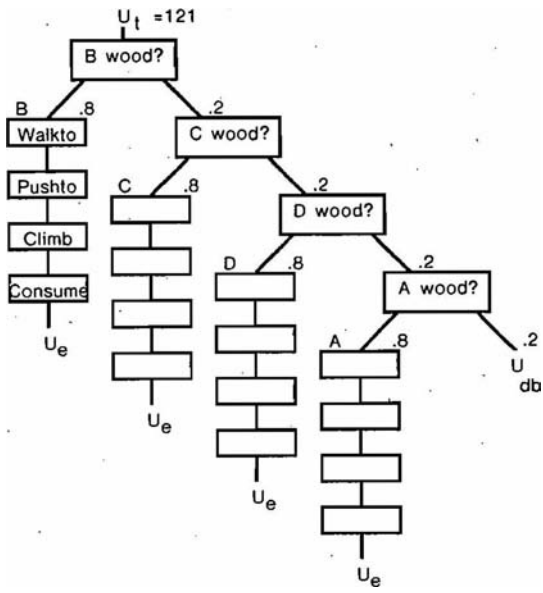

FIG. 5 Inserting costless tests for wooden boxes.

A plan can be improved by developing plans to recover from failures in the original outline. The failure in Fig. 3 can be elaborated with steps to clear away the mess, choose an alternative box, and try to use it to reach the bananas. Such an elaboration is shown in Fig. 4; the expected utility of the plan has risen from 87 to 96 as a result of the elaboration. The increase occurs because the plan to deal with the failure of box B (i.e., to try again with box C) has a higher utility than that of giving up (45 vs. 0). This process can be carried on indefinitely, but if the probability of failure is fairly low, the cost of additional planning may exceed the slight improvement in expected utility. (Generating plans for recovering from failures is similar to the generation of the original plan: a symbolic problem solver can provide plan outlines; the alternative recovery strategies are compared with utility measurements.)

Elaborating failures tightens the bounds on a plan. Figure 4 has a set of bounds shown in brackets as a triple: lower bound, expected value, and upper bound. Bounds are assessed from bottom to top; the triple with a prime symbol is calculated using the U_u formula, then the effects are propagated up through the tree. This process yields a rather tight bound on the utility of using box B (cf. Table 2).

Another kind of plan improvement can be achieved by introducing steps in the plan to gather information, and thereby reduce the uncertainty in the outcome. A simple example is shown in Fig. 5: a perfect and costless test determines whether each box is wooden. If the test announces that a box is wooden, which happens

with probability p_0, then the plan to use that box is guaranteed to be successful. If the test announces that a box is cardboard, the next best plan is tried, and so forth. Adding these tests causes U_{total} to rise to 121.

A more realistic model of such information gathering incorporates the expenditure of resources required to perform the test and for the possibility that the test gives an incorrect answer. We shall define two such tests that can be used to elaborate the monkey-and-bananas plan:

TEST-FAR: A visual test measures whether a box is wooden. It does not require that the monkey be located near the box. It has a cost C_{tf}. The answer is characterized by two conditional probabilities p_{fw} and p_{fc}.

$p_{\text{fw}} = \Pr\{\text{test announces ``wood'' } | \text{ box is wooden}\}$
$(1-p_{\text{fw}} = \Pr\{\text{test announces ``cardboard'' } | \text{ box is wooden}\})$

$p_{\text{fc}} = \Pr\{\text{test announces ``wood'' } | \text{box is cardboard}\}$
$(1-p_{\text{fc}} = \Pr\{\text{test announces ``cardboard'' } | \text{ box is cardboard}\})$

If the test always yields correct answers, $p_{\text{fw}} = 1$ and $p_{\text{fc}} = 0$.

TEST-NEAR: This test is analogous to TEST-FAR, but the monkey must be at the same location as the box being tested. This test might involve ''thumping'' the box. Cost C_{tn}. The behavior is characterized by:

$p_{\text{nw}} = \Pr\{\text{test announces ``wood'' } | \text{ box is wooden}\}$

$p_{\text{nc}} = \Pr\{\text{test announces ``wood'' } | \text{ box is cardboard}\}$

If the test always yields correct answers $p_{\text{nw}} = 1$ and $p_{\text{nc}} = 0$.

Adding these tests to the plan outline produces the four strategies shown in Fig. 6. In order to calculate the expected utilities of these plans, and thereby choose the best one, we must describe the consequences of performing a test. We shall use a simple Bayesian model: a test causes a change in the probability that the tested box is wooden, according to Bayes rule:

$$\Pr\{\text{box is wood} \mid \text{TEST-FAR announces ``wood''}\} = \frac{p_{\text{fw}} \Pr\{\text{box is wood}\}}{p_{\text{fw}} \Pr\{\text{box is wood}\} + p_{\text{fc}} (1-\Pr\{\text{box is wood}\})}$$

$$\Pr\{\text{box is wood} \mid \text{TEST-FAR announces ``cardboard''}\} = \frac{(1-p_{\text{fw}}) \Pr\{\text{box is wood}\}}{(1-p_{\text{fw}}) \Pr\{\text{box is wood}\} + (1-p_{\text{fc}}) (1-\Pr\{\text{box is wood}\})}$$

In these equations, $\Pr\{\text{box is wood}\}$ is the *prior probability* that the box is wood, and $\Pr\{\text{box is wood} \mid \text{TEST-FAR}\}$ is the *posterior probability* that it is wood. Analogous relations hold for TEST-NEAR.

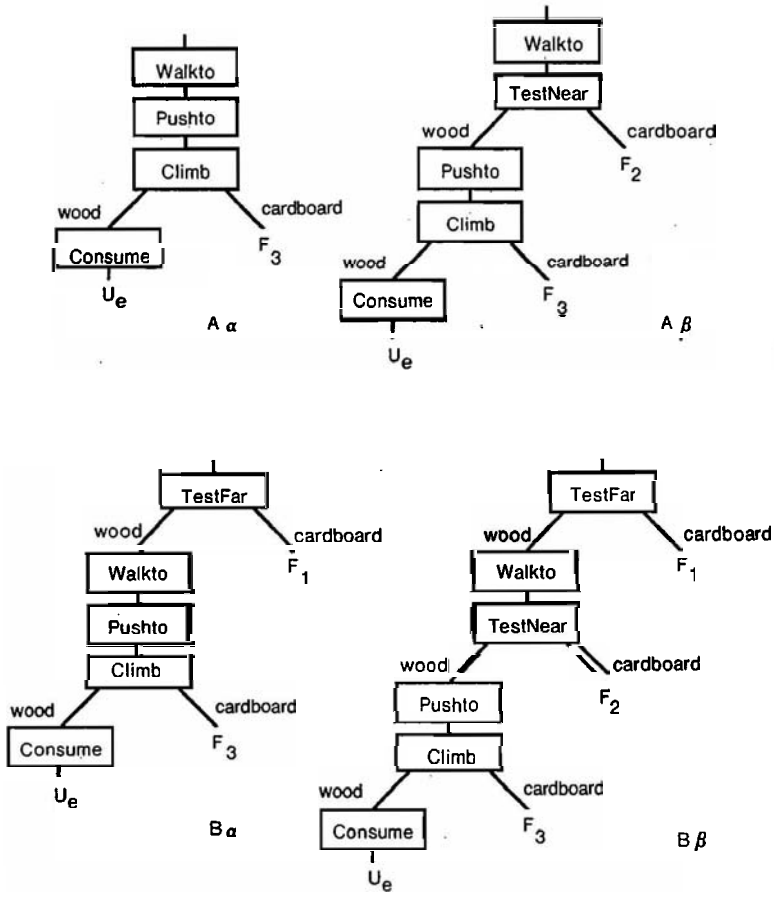

FIG. 6 Four strategies using TEST-NEAR and TEST-FAR.

We also need to calculate the probabilities of taking each of the two paths that emanate from the TEST operation:

$$\Pr\{\text{test announces ``wood''}\} = p_{\text{fw}} \Pr\{\text{box is wood}\} + p_{\text{fc}} (1 - \Pr\{\text{box is wood}\})$$

$$\Pr\{\text{test announces ``cardboard''}\} = 1 - \Pr\{\text{test announces ``wood''}\}$$

As an example of these calculations, we shall evaluate the expected utility of the Bα strategy of Fig. 6 applied to box B, with the prior probability of finding a wooden box, p_0, set to 0.8, the performance of TEST-FAR characterized by $C_{\text{tf}} = -20$, $p_{\text{fw}} = 0.9$, and $p_{\text{fc}} = 0.1$, and the failure utilities U_{F1} and U_{F2} set to 0.

For each of the three paths through the tree, we must calculate the probability the path is taken and the utility of the path:

Path	U_i	*Path probability*
TEST-FAR, F_1	−20	Pr{test announces "cardboard"} $= 1-(p_{tw}p_o + p_{tc}(1-p_o)) = .26$
TEST-FAR, WALKTO, PUSHTO, CLIMB, F_3	−89	Pr{box is cardboard | test announces "wood"}* Pr{test announces "wood"} $= [1-(p_{tw}p_o)/(p_{tw}p_o + p_{tc}(1-p_o))]* .74 = .02$
TEST-FAR, WALKTO, PUSHTO, CLIMB, CONSUME, U_e	106	Pr{box is wood | test announces "wood"} * Pr{test anounces "wood"} $= [(p_{tw}p_o)/(p_{tw}p_o + p_{tc}(1-p_o))] * .74 = .72$

The expected utility is $EU = \Sigma p_i U_i = 69$. Similar calculations for all four strategies, applied to box B, are recorded in Table 3. For the given set of costs, utilities, and probabilities, strategy Aβ is selected. The strategy can be improved still further by elaboration to cope with the failures F_2 and F_3, as described above. Using both methods, a strategy with expected utility 105 turns out to be optimal.

The model of testing reveals tradeoffs among various information-gathering strategies as differences in utility. If the insertion of tests in a plan causes the expected utility of a plan to rise, the test is providing information that helps reduce the uncertainty of the outcome. Decision theory calls the increase in utility the "value of information."

If different strategies have nearly identical utilities, as do Aα and Aβ in Table 3, the planner might announce indifference between the strategies, and perhaps use other methods to decide which one to pursue. Such small differences may be insignificant when uncertainties in the probability or utility models are taken into

TABLE 3

Strategy	*E U*
Aα	87 (cf. Fig. 3)
Aβ	88
Bα	69
Bβ	63
$C_w = -1$ * distance	$C_{tf} = -20$
$C_p = -10$ * distance	$C_{tn} = -10$
$C_b = -20$	$p_{tw} = .9$
$C_c = -5$	$p_{tc} = .1$
$U_e = 200$	$p_{nw} = 1$
$U_{F_t} = 0$	$p_{nc} = 0$
	$p_o = .8$

account. Although we may in principle reduce these errors by refining the model, we shall always be faced with insignificantly small differences.

Elaborations cause the search space to grow quite large because of the various choices of inclusion and exclusion of tests, the increased number of failures that require recovery strategies, etc. The search would be wholly impractical without a guide such as the branch-and-bound algorithm. We shall address below other methods of combating the "combinatorial explosion" during elaboration.

4. WORLD MODEL ACQUISITION

The planning activities described in previous sections have assumed that the planner has a complete model of the world. Because acquiring such a "world model" and locating all the boxes is a sizable task, an efficient strategy for feeding the monkey must make efficient allocation of resources to build the model.

A decision-theoretic model of the acquisition process can express the cost and reliability of an acquisition operator and the utility and probability of locating an object in the world. Once again, the utility measure can be used to search for an efficient strategy. A key concept in this approach is that the expected utility of a plan that uses an object, as computed in Section 2, can be used to estimate the value of locating the object.

The vision strategy must decide where to look. For our example, we shall use a grid to divide the world into regions and use a utility calculation to decide which region should be scrutinized. We shall use a simple acquisition operator LOOKAT:

LOOKAT(x,y). Examine the unit square at (x,y) with a vision system to determine if a box lies in the square. The cost of the operator is $C_{x,y}$. The outcome of the operator could be characterized by the two probabilities:

$p_{\text{lb}} = \Pr\{\text{LOOKAT}(x,y) \text{ announces "box"} \mid \text{box at } (x,y)\}$

$p_{\text{ln}} = \Pr\{\text{LOOKAT}(x,y) \text{ announces "box"} \mid \text{no box at } (x,y)\}$

In the remainder of the example, we shall assume $p_{\text{lb}} = 1$ and $p_{\text{ln}} = 0$.

In addition, we shall require a priori estimates of the probability that a box lies in a square, $\Pr\{\text{box at } (x,y)\}$. The utility of looking at a square is thus:

$$U_{\text{look},x,y} = C_{x,y} + \Pr\{\text{box at } (x,y)\}\, U_{\text{box},x,y} + (1 - \Pr\{\text{box at } (x,y)\})\, U_{\text{fail}}$$

where $U_{\text{box},x,y}$ is the utility of using a box found at square (x,y), which is estimated by evaluating the utility of a plan outline (e.g., Fig. 3) without elaboration. The $U_{\text{look},x,y}$ values are calculated for all squares, and the square with the largest value is chosen as the best place to look for a box.

The vision plan can also be elaborated. If the LOOKAT operator fails to locate a box, we might apply the LOOKAT operator to another square, and so forth (Fig. 7). This is just like coping with failure in CLIMB—we chose an alternative. Calculating the order in which to look at squares in this case is straightforward: the square with the largest value of $U_{box,x,y} + C_{x,y} / \Pr\{\text{box at } (x,y)\}$ is examined first, that with second largest next, etc.

Analogous to adding TEST-NEAR and TEST-FAR operators might be adding CHANGE-LENS or TURN-ON-MORE-LIGHTS operations to improve the probability of success of the vision operator. A more subtle elaboration might include WALKTO operators to position the monkey for better viewing of a particular region; this would of course influence the values of $U_{box,x,y}$ by changing the initial position of the monkey.

The utility of an acquisition plan is an attempt to measure the benefit of augmenting the information in the world model. Finding objects may permit using plans of higher utility; or it may show that a navigational path is blocked. Absence of objects may be useful when planning a collision-free route. Hence the utility that is assigned to a particular outcome of a vision operator is the utility of the information for the current set of plans. The utility of acquisition is further increased because new world-model information may give rise to alternative plans not previously considered.

The results of the LOOKAT operation change information in the world model. If a box is located, it is recorded in the model. In all cases, the subjective probability that a box is located in the scrutinized square, $\Pr\{\text{box at } (x,y)\}$, is modified. This is analogous to the treatment of TEST-NEAR and TEST-FAR: Bayes' rule is used to update $\Pr\{\text{box at } (x,y)\}$ just as it is used to update $\Pr\{\text{box is wood}\}$ as a result of the TESTs. This means that once a square is looked at and found not to contain a box, it will probably not be tested again.

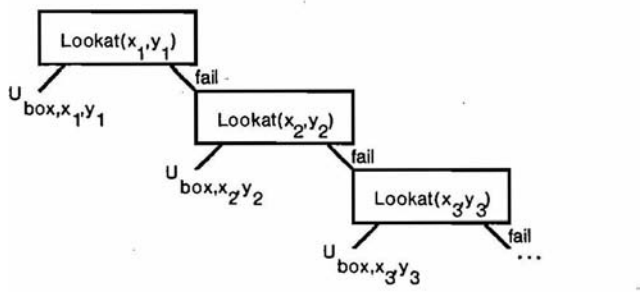

FIG. 7 Correcting failures in LOOKAT by examining more squares.

The a priori values for the Pr{box at (x,y)} are supplied by a function that can contain considerable information about the world. If boxes are more common in the garage than in the house, this can be expressed in the probability assessments.

The results of looking at a square may have implications other than the success or failure in locating a box. An object needed to execute a competing plan may be located, thereby causing the utility of such a plan to rise. Or the information detected by LOOKAT may alter probabilities that boxes exist in surrounding squares (e.g., finding a fireplace may cause one to suspect that no boxes lie nearby; if piles of boxes are common, finding a box may increase chances of finding others nearby).

The previous discussion suggests that we look for only one box, and then use it. However, once a single box has been located, we can consider alternatives: use the one that is found or look for another one. If a second box is located, it may have higher utility than the first, or may improve the plan to use the first box because the second is available as backup.

Because acquisition operators change the world model, the results can cause widespread changes to the utilities of current plans. We could, in principle, model an acquisition operator with a large number of outcomes and generate plans for each contingency although a large number of eventually useless plans would result. A mechanism to control the amount of planning ahead and to permit periodic reevaluation of plans is clearly needed. The next section addresses this topic.

5. THE TRINITY: LOOK, THINK, ACT

At some point, the planning operations sketched in the previous sections must be halted and the best plan actually executed. In fact, planning must be severely limited, lest planning resources be wasted in any of numerous ways, such as generating detailed plans for paths that are never encountered or planning without adequate world model information or pursuing complicated elaborations that increase plan utilities only slightly. However, if planning is curtailed, we must be able to resume planning later on.

What is needed is an efficient scheduling of planning, looking, and acting. The scheduler decides in some way which activity is most beneficial at the moment, grants it a resource quantum, and then repeats. A natural quantum for looking and acting is execution of one of the "operators" such as LOOKAT or WALKTO. A natural quantum for planning might be one iteration of a branch-and-bound algorithm, or the addition of one elaboration to a plan.

The decision to plan or to execute can be made with a utility measure. We compare the utility of looking (i.e., executing a step in the best acquisition plan), acting (i.e., executing a step in the best action plan), or additional planning (i.e.,

elaborating existing plans with branch-and-bound as a guide, or developing more symbolic plans). Unfortunately, specifying a utility function that reflects the benefits of future planning is quite difficult.

Decision theorists have addressed a problem called "cost of analysis," which is loosely related to the notion of planning cost used here (Matheson, 1968). In a practical analysis, the cost of building a model and assessing probability and utility values is often large enough to try to estimate the value of various alternative analyses. This calculation is often modeled as a set of initial tests of varying cost that give different sorts of information about prevailing probability distributions. The tests and their costs correspond to the various analysis choices (Matheson, 1968).

In our case, planning is the application of the model to a particular situation, which may involve substantial symbolic reasoning, tree expansion, etc. We desire simply to discount the value of a partial plan by the cost of the processing required to generate the details needed for execution.

A simple ad hoc approach can be used to *limit* planning activities, that is, to specify a stopping criterion. The difference between the upper bound and expected utilities of a plan for vision or action is a limit on the improvement in the plan that infinite planning would achieve. We have the choice between executing the plan as it stands, and receiving (on the average) the expected utility, and spending some effort planning (say, the cost is C_{plan}) and receiving, *at most*, the upper bound on the plan. Comparing the utilities of these two alternatives, we have:

Execute: (U_{execute})	$U_{\text{execute}} = U_{\text{plan expectation}}$
Plan with cost C_{plan}: (U_{plan})	$U_{\text{plan}} \leq C_{\text{plan}} + U_{\text{upper bound}}$

If $U_{\text{plan}} > U_{\text{execute}}$, we choose to plan. This constrains the planning effort: $C_{\text{plan}} \geq U_{\text{plan expectation}} - U_{\text{upper bound}}$. Thus the additional planning effort is limited by the difference between the upper bound and the expected utility. Obviously this is a crude approximation and could be refined.

This approach essentially compares the risk of the current plan (as estimated by the difference between the upper bound and the expectation) with the cost of further planning. It does not attempt to predict the actual value of planning, but rather measures the cost and maximum value of planning steps. It would certainly be better to use the expected value of the benefits of planning if this quantity could be computed.

The main loop of the system plans until such a stopping criterion is reached, and then either looks or acts, whichever has the greater utility. Then the process repeats. The outcomes of looking or acting are, of course, recorded and cause

adjustments in the utilities of various available plans. This mixing of planning and acting is a uniform framework for providing "monitoring" and "verification" functions in robotics systems (Grape, 1973; Munson, 1971).

6. BEYOND THE EXAMPLE

The monkey-and-bananas example used in the preceding four sections fails to bring out some of the potential uses of decision theory. This section expands on the techniques illustrated in the example.

The Model

The design of the abstracted model of the world, comprising the symbolic operators, the outcomes of operators, and the probability and utility assessments, is a key to the performance of the system. On the one hand, we contend that a repertoire of symbolic operators alone, as used in many robotics systems, generates plans that are blatantly wasteful of resources. The augmented operators described in the monkey-and-bananas example, which include cost and reliability measures, still fall far short of a complete model of "reality." Yet if the model gets too complicated, with many possible elaborations, outcomes, and failures, the search may grow unmanageably large.

Modeling the Operators. The cost/outcome model of system operators strives to summarize, in a few functions, the behavior of large, complicated subsystems for vision or action. Surely the complexity cannot be captured in a few simple functions. On the other hand, an excessively precise model of the operation of the subsystems would paralyze planning, turning it into a huge simulation. A numerical summary form for expressing performance of subsystems for vision or action which permits computation on the bounds and estimates (expectations) of performance is quite powerful.

Symbolic Models. The symbolic model gives rise to plan outlines that constrain the remaining search to *reasonable* plans. In our example, the symbolic model will not generate a plan to fly to the box; flying is not reasonable. Thus, utility analysis is not applied to arbitrary sequences of operators which violate logical conditions (i.e., sequences in which the preconditions for each step are not met) or which are not reasonably likely to achieve the given goal.

Utility Functions. The utility function must capture the tradeoffs the planning system is expected to make among reasonable plans. In the monkey-and-bananas example, we assign simple cost functions to each operator and a fixed utility to being fed. This permits tradeoffs among boxes at different locations to be expressed. However, the linear additive property of our utility function (the

total utility is a simple sum of independent contributions of the operators) cannot express certain preferences, especially aversion to risk.

The utility calculation can be modified to be more comprehensive. A trivial extension will express risk aversion: Let the utility of a plan be $U = f(\Sigma C_i + U_o)$, where the C_i are the costs of the steps in the plan, U_o is the contribution of the outcome, and f is a convex monotonically increasing function (see, for example, Raiffa, 1970, for an explanation of how this construction models risk). Further extension permits the costs to be interdependent; the utility is then a function of all steps in the plan.

Because the utility function will be evaluated frequently during the search and elaboration processes, the expense of its computation is important. An additive property permits the calculation of the utility of a partial path to be local and incremental, rather than global. Even if the utility of a traveler, for example, is a nonlinear function of total transit time and total fares, running totals of elapsed time and spent money can be kept with each node, and the nonlinear utility function f can be computed locally. In this case, each operator is characterized by a cost vector $\mathbf{C}_i$ which expresses the resources required to accomplish the step (e.g., time, energy, money, etc.). Then the total utility of a path can be written as $g(\Sigma \mathbf{C}_i + \mathbf{U}_o)$.

The information that must be associated with each node of the decision tree is often local and compact, as this example indicates. A few numbers for utility calculations, probability of reaching the node, and a small amount of information about state changes caused by the operator at the node suffice. If the state information needed by the symbolic problem solver can also be represented compactly, a system will be able to explore many alternatives without excessive space requirements.

Vision and Other Acquisition Operators. Historically, the vision portions of a robotics experiment have been the most complex, the most consuming of resources, and the most failure-prone of the entire system. Similarly, in a medical domain, the cost and risk of gathering information can be quite high. Consequently, we desire that a planner generate reasonable strategies for acquisition. For the purposes of planning a good overall strategy, we must strike a balance between a "black box" model of vision subsystems and a detailed model of their performance.

The important point is that a utility function is a natural way to compare differing information-gathering strategies, to measure the benefits of improving them, and so forth. We do not attempt here to expand the repertoire of basic vision techniques, but rather to provide a framework in which good vision strategies can be planned. Within the vision subsystems, additional planning will certainly occur, perhaps generating symbolic plans (e.g., acquire the chair by first scanning vertically for the chair back, then look for the seat, Garvey & Tenenbaum, 1974), or perhaps using theories of optimal testing (e.g., sequential decision theory, see deGroot, 1970; Bolles, 1976).

1. How to decide what to look for. The utilities of competing plans provide a measure of which objects are important to the system. These measures can propagate to the lowest level of a vision subsystem, appearing as preferences of, for example, operators to locate corners or operators to measure texture. The task of assimilating low-level vision information and "recognizing" objects may also make efficient use of such information.

2. How to decide when to look. When should vision operators be invoked to locate objects needed in a particular plan? The elaboration process can apply vision operators both earlier and later, using the expected utility to decide whether improvements result. Two simple examples illustrate the need for careful consideration of when to look:

> A block-stacking task requires two blocks; both are located with vision before the manipulator is used. Because one block hides another from the camera's viewpoint, locating the second block is extremely costly. If the arm had moved the first block away to a stack, the second would *then* be found easily.
>
> A block-stacking task requires two blocks; one is located, and moved to the stack. The attempt to locate a second block fails completely—there is not one available. The goal is unsatisfiable, and could have been abandoned before manipulation effort was expended.

These simple scenarios suggest that it may be advantageous to acquire a complete world model before planning. First, as illustrated in the second example, failure to find enough parts to satisfy a goal causes failure early, which is more efficient than detecting the failure later. Second, more information often permits cleverer plans: if many blocks have been located, an optimal assignment of blocks to locations in the stack can be made (Fahlman, 1974). Third, there are often economies in making measurements in parallel. In robotics domains, this may mean that the TV camera need be read only once; or it permits planning aperture, orientation, and other parameter settings that can apply to groups of visual inquiries. (If the strategy is laid out as in Fig. 7, we can compute the expected number of LOOKAT operations required, and set the camera viewing parameters to look at the spots specified in the first LOOKATs.) In medical diagnosis, making parallel measurements may shorten hospital stays or reduce the number of costly preparations for tests (Ginsberg, 1969).

There are also advantages in delaying vision. First, better observation conditions may exist after several execution steps have been performed, as illustrated above. Second, the omnipresent opportunity to destroy information argues for delaying its gathering: if the arm drops a block on a collection of carefully located objects, the location effort is largely wasted. Third, late vision may benefit from additional world-model information which may help decide where to look.

The utility measure can express the tradeoffs between these two extremes. If, for example, toothpicks occur rarely in the world, a plan requiring a toothpick

may be improved by locating the toothpick early. Similarly, if a particularly error-prone step may destroy information, late information gathering might be preferred. The elaboration process considers moving these vision operations to early positions in the plan; it uses the utility measure to decide whether the change is advantageous.

3. How to decide where to look. The main techniques used to decide where to look were discussed in the example: a probability density function and utility function on objects in the world together specify where to look. A great deal of information, both a priori information about the world and the results of previous tests and actions, can be expressed in the probabilities. The particular formulation given in the example has the virtue that the choice of squares and examination sequence has a very simple solution that requires no search. The operations research literature contains analyses of methods for searching for objects that also yield such closed-form solutions (Cozzolino, 1972).

Bayesian Models. In the monkey-and-bananas example, results of tests are modeled with Bayes' rule: a posterior probability is computed as a function of the prior probability and some properties of the test.

One problem with this technique is that it makes the often unwarranted assumption that the tests are independent. Strategy Bβ of Fig. 6 has a path that applies both TEST-FAR and TEST-NEAR. In this case, the independence assumption may be violated: if both tests are visual, they may be prone to similar errors in, say, dimly lit scenes, and hence have correlated results. In principle, this problem can be resolved by using different values for the performance of TEST-NEAR (i.e. p_{nw} and p_{nc}) if the test follows a TEST-FAR on the same box, although quantifying the dependencies may be difficult.

If the probability and utility models are extended to their most general form, in which all quantities are modeled as distributions rather than as discrete values, the expressive power of the model is enhanced. For example, Bayes' rule for the LOOKAT operator, which modifies a distribution function on the position of a box, can record the uncertainty in the location of the box. This information may be useful to calculate probabilities of success of subsequent operators. For example, the success of a GRASP operator in picking up a box with a mechanical arm depends on the precision with which the location of the box is known; the stability of a stack of boxes depends on the precision with which they are located, picked up, and stacked (Taylor, 1976). If the success of the plan is sufficiently impaired by imprecise location information, elaboration may find it beneficial to insert more precise tests that reduce the uncertainty. A simple representation of a distribution function (e.g., mean and standard deviation) may suffice for these calculations (Sproull, 1977).

Modeling Execution Operators. The planning system views "execution" as calling upon a vision or action subsystem to perform a particular task, and then recording the results. However, it may happen that the outcome of the subsys-

tem's activity is not known precisely. For example, if the subsystem is to turn on lights, we may not know whether the lights work. Elaboration can decide between two approaches by computing their utilities: either continuing obliviously, hoping that the operator worked correctly, or applying some test that is capable of determining the state of nature. If an operator were inserted by elaboration to turn on lights, the utility of the original plan without lights is known, and can be used to decide whether a test that verifies the strength of illumination increases the expected utility.

The Role of the Symbolic Problem Solver

The chief function of the symbolic problem solver is to provide plan outlines for elaboration and execution. Ideally, planning occurs at various levels of detail: the initial planning is done in a highly abstracted model of the world, with an associated abstracted utility model so that estimates of the utility of the plan can be made, thus permitting comparison of competing goals. Further planning is accomplished using more detailed models. (This technique is drawn from current symbolic problem solvers, see Sacerdoti, 1974, 1975; Sproull, 1977.)

Generating the initial plan outlines may require no search at all: a system may use so few outlines or the outlines may be so simple that they are stored as "constants" and simply retrieved for purposes of elaboration and execution. This technique is more attractive if plan outlines may contain variables (for example, the object BOX in our initial plan outline could be a variable); the symbolic problem solver instantiates an outline for each possible application. The outlines for the monkey-and-bananas problem, the wheelless student, and even for block-stacking tasks (see Fig. 8) are of this simple nature.

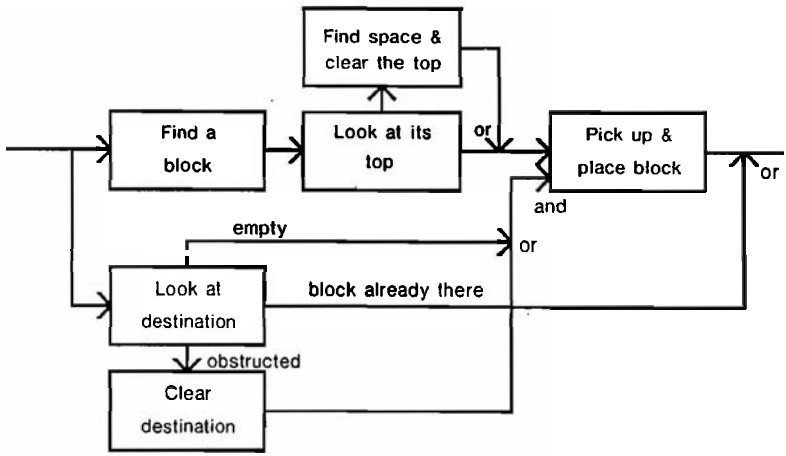

FIG. 8 A plan outline for block stacking.

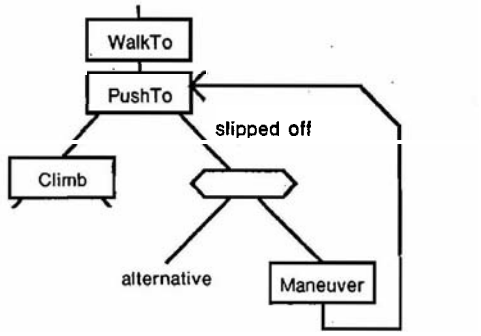

FIG. 9 A recovery strategy.

The plan outlines delivered to the elaboration process require additional symbolic information. Some is "state" information that is used to match preconditions of various elaborations (e.g., TEST-NEAR can be inserted only when the monkey and box are at the same location). Some is "recovery" information, specifying how a recovery from a particular error may be accomplished. We have already demonstrated the simplest form of this: on failure, choose another top-level alternative. More complicated error recoveries try to make a fix-up and then join the original plan again. This procedure changes the decision tree into a *graph,* and permits a somewhat more efficient search; joining an existing plan saves having to generate a fresh one. The problem solver can generate, for each outcome of an operator, one or more reasonable failure-recovery approaches. Then the elaboration process will calculate which of the approaches is best (see Fig. 9).

Elaborating

The elaboration process, which explores improvements to plan outlines using the utility function as a measure of progress, includes the following operations:

1. Fixing failures. Paths in the plan outline that end in failure are expanded to recover. Often, this involves pursuing another top-level alternative plan. In this case, an estimate of the utility of fixing up the failure is the current utility assessment of the top-level alternative. This is, of course, not completely correct, because the state of the world used to compute the top-level utility is not the same as that after a failure. For example, in the monkey-and-bananas problem, failure F_3 (Fig. 6) leaves the monkey under the bananas, and the ruins of a cardboard box under the monkey. Later, if the search for good strategies indicates effort should be devoted to this plan, the failure elaboration may be improved from an estimate to an explicit plan.

2. Inserting steps. The insertion of tests has already been demonstrated; the location of such insertions is governed by preconditions on the test and state information provided in the plan outline. The elaboration process only considers

inserting tests that will affect the outcome of subsequent steps, that is, tests that modify a parameter used to calculate the cost or outcome probabilities of a subsequent step. Determining how to insert tests that change the basic plan outline by changing the state of the world (e.g., tests that require the monkey to make additional moves for a better view) is very difficult.

3. Changing operators. The model may provide several operators that accomplish the same operation (from the standpoint of the symbolic model) but with different costs, or different reliabilities, etc. Thus we might have two operators: WALKTO and WALKTO-AVOIDING-OBSTACLES. The choice of operator (or subgraph of the plan outline) is controlled by utility appraisals.

4. Moving operators. If a plan outline is a sequential union of several outlines (e.g., stacking two blocks, the QA4 "buy groceries and mail a letter" problem (Rulifson, Derksen, & Waldinger, 1972), or certain assembly problems (Taylor, 1976)) it may be advantageous to reorder some of the steps. A simple case, the grouping of vision operations, was discussed above. In general, however, this is a very hard problem. The decision-theory techniques provide a useful way to decide if progress is being made, but they do not obviate a considerable amount of symbolic reasoning to decide whether the plan outline remains legal.

The elaboration process has a strong parallel with trial evaluation: a plan modification is tentatively made, the utility of the new plan is computed, and the modification is saved if the utility rises. Thus a test will be inserted if its "value of information" is greater than its cost because the recalculation of the utility automatically incorporates both of these influences.

The elaboration process needs heuristics in order to keep the search reasonably efficient. The monkey-and-bananas example shows such an effect: the vision planning uses an estimate of the utility of actually using a box; the execution planning uses an estimate of the utility of locating and using a box that is currently unavailable. Figure 10 shows such a situation schematically: the utility

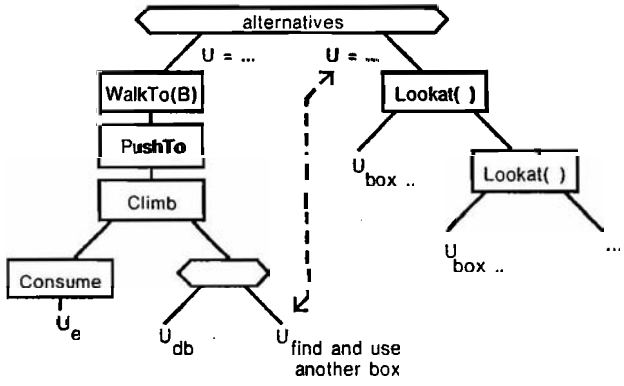

FIG. 10 Estimating a recovery utility.

of the best vision plan is used as an estimate of the utility of a recovery strategy that uses an as yet to be located alternative box; the vision calculation uses an unelaborated utility of using a box (in conjunction with any that already exist) should it be found. Thus both the vision and action planning activities each make use of the other's approximate utilities.

Looking, Thinking, Acting

The allocation of effort among the various activities of the system, looking, planning, and acting, can be thought of as a scheduler, allocating resources to the task of highest priority. Priority is, of course, determined by a utility measure. This approach has several implications for the design of the system.

First, planning is incremental. After a vision or action step, the results are recorded, perhaps causing changes in the utility or probability assignments of pending plans, and planning is continued. Since *every* alternative plan is in some sense "active" (i.e., the planning is complete to some level of detail, at least enough to estimate a utility to compare with other plans), each outcome can cause the planner to "reexamine the alternatives," and perhaps adopt an entirely different strategy. The vision routines may also take advantage of the incremental organization, using latest utility estimates of objects needed to plan a vision strategy.

Second, the formulation helps prevent needless detailed advance planning. Some elaborations consume planning effort, although their contribution to increasing plan utility is slight; the effort is weighted by the probability of reaching a particular part of the tree. As execution progresses, however, some of these alternatives will vanish (paths not taken) and some utilities will increase, thus making detailed planning more attractive. The planning horizon at any level of abstraction thus precedes the execution in a controlled fashion. This property can be exploited to delay certain specific planning activities. An example from robot block stacking is the freespace problem: if a free spot is needed in which to place an object temporarily, it is advantageous to delay assigning the location until absolutely necessary; it may be easy to examine available alternatives at the time the space is needed.

Our simple formulation of the utility of planning can undoubtedly be considerably improved. Ideally, various intuitive measures of planning progress, such as the number of alternative branches remaining to be explored, or estimates of what clever but costly planning tricks may accomplish, could be included in the utility measure. This measure should provide ample rewards for carefully assessing the economics of planning.

7. THE PARADIGM

This paper advances the view that a combination of decision-theoretic and symbolic artificial intelligence paradigms offers advantages not available to either individually. This section explores this claim.

The Two Fields

Some readers will have already objected that our suggestions do not increase the range of problems solvable by decision theory or symbolic processing, that each is a powerful and complete paradigm, and that our remarks bear on efficiency considerations alone. The pure symbolic processor claims that he can achieve the effects we describe by dividing numeric ranges into a small number of "symbolic values" (e.g., temperature into COLD, COOL, WARM and HOT) that suffice for a given problem. Information about tradeoffs can be encoded as a set of symbolic preferences: (FED and WALKED-A-LONG-DISTANCE) is preferred to ((not FED) and WALKED-A-SHORT-DISTANCE). Or he will assess tradeoffs numerically by instantiating theorems of number theory, analysis, and algebra. This gives rise to crude and awkward models in cases where a small amount of numerical processing is more natural and accurate.

The pure mathematical programmer, on the other hand, will mathematize all constraints or move complexity into value or reward functions. He will formulate any search as a shortest path problem with appropriate arc weights and propose dynamic programming to calculate a solution. The result is often a huge state space for very simple problems, making numerical solution simply infeasible.

Practitioners of either field adopt more moderate approaches: the AI designer finds many problems suited to partially numerical approaches. Similarly, the decision theorist engages in a substantial amount of symbolic reasoning to formulate his model and to apply it intelligently to the situation; he may also use "heuristic" solution techniques on large problems. A human analyst will perform the reasoning required to build a decision tree intelligently, one that represents sensible plans. From the point of view of AI, this construction process is itself an endeavor of interest.

From the point of view of decision theory, our formulation aims to permit a computer program to emulate a good decision analyst. Such an analyst combines formulating plans and searching decision trees to arrive at a solution. A good analyst will monitor the implementation of the decisions, keeping abreast of exogenous changes in the utilities on which his solution was based, formulating additional plans, etc. Our technique attempts to emulate this activity. This is in contrast to conventional computer programs used to search one static tree exhaustively.

From the point of view of AI, the advantage of decision theory is the ability to find solutions that are "optimal" in some model. Although the approach requires a certain amount of search to find solutions, several powerful methods are available to limit the search.

1. The symbolic problem solver constrains the search later undertaken to perfect a strategy. The elaboration search does not try strategies involving many combinations of many operators, but is limited to those strategies that include certain key steps specified by the problem solver. The basic recovery strategies,

that is, the instructions for plan elaboration in case of failure, are also provided by the symbolic problem solver. Information attached to the plan outline is used to guide plan elaboration. For example, tests which have certain preconditions (e.g., that the monkey is "at" a box) are considered only at points in the plan outline that meet the conditions. A rough plan generated in simplified, abstract space, can be used to constrain the more careful planning (Sacerdoti, 1974, 1975). These are examples of basic search-limiting methods of AI not practiced in decision analysis programs.

2. A number of decision-theoretic techniques limit search. Branch-and-bound methods limit search based on bounds derived from the utility models. In addition, one can *prove* that the failure F_1 in Fig. 6 should not include paths that persist in using the same box (i.e., paths that disregard the outcome of the test): every such strategy is dominated by one that simply does not perform the test at all. Such "utility theorems" limit search.

Another example of search limiting occurs when the plan outline specifies a loop. Any paths that involve loops continue to incur increased costs as they are expanded, but the ultimate utility is fixed. The loop thus expends effort without approaching the goal; such paths will be cut off by the branch-and-bound algorithm.

3. Domain-independent heuristics can be applied to limit search. One such heuristic is to explore paths of high probability first, and perhaps be willing to bound pessimistically those paths of low probability. Although pure decision theory looks dimly on this technique because even paths of low probability may have unbounded utilities, in many cases we can meaningfully assign bounds to the utilities.

Certain of the recovery mechanisms, for example, using another top-level alternative, are domain independent, as is the method of approximating the utility of such an alternative plan.

4. Domain-dependent heuristics can limit search. Although these techniques may require a certain amount of reprogramming for each new domain, they are probably far more powerful than domain-independent methods. The current AI trend toward knowledge-based systems (Bobrow & Collins, 1975; Fikes, 1976; Nilsson, 1974) is due in part to benefits of distributing domain knowledge throughout systems. Such techniques are equally applicable in our framework.

Search-limiting heuristics are not without drawback—the resulting search may not guarantee finding the optimal solution, that is, it is not admissible. However, the utility measure still allows us to extract the best plan among those developed in the search.

The Combination

What the two fields of decision theory and artificial intelligence offer is a collection of techniques that can be applied judiciously to solve problems. There are cases when applying decision theory is difficult or adds little to AI techniques:

Insignificant costs. The benefit of optimal planning may simply be too low if the costs of the planning and execution are themselves insignificant or if the planning costs greatly exceed the execution costs.

Identical values. A problem may give rise to solutions of identical preference. A theorem-proving program may not be at all concerned with finding the shortest proof or with the expense of the search. A program that attempts to "understand" a paragraph of natural language in order to answer questions about it is likewise not concerned with optimization but with capturing a conceptual structure. In these cases, the utility function on outcomes is nearly constant, and gives no information to the search.

Both of these examples are characterized by the intuition that the domain is inherently symbolic: the understanding problem is to build a conceptual structure that is communicated as a string of words; the theorem-proving task, even as practiced by humans, is primarily symbolic manipulation. There are notions of "best" solutions in both cases, but they are second-order considerations.

Partial plans. Classical decision theory deals only with complete plans, and cannot cope with extremely large search problems in which it is impractical to enumerate all plans (e.g., chess, with 10^{160} nodes). Some AI programs deal with this problem by using a heuristic estimate of the value of a partial plan to approximate the value of a complete plan (Newell, Shaw, & Simon, 1958).

The heuristic estimate is often "backed up" through several steps of the partial plan in order to calculate the utility of the entire plan. This technique reduces somewhat the sensitivity of the overall utility calculation to errors in the heuristic estimate. In spite of this reduction, errors in the utility are a major concern (Pohl. 1973). It is sometimes possible for a program to "learn" which estimates give rise to the smallest errors (Samuel, 1967).

Modeling difficulties. It may be very difficult to construct a utility and probability model that applies to the problem. Although the central theorem of decision theory shows that any choice of a "best" plan is an implied assessment of utilities and probabilities, it still may be difficult to cast the model in numerical terms.

A particularly painful aspect of this problem is presented by Bayes' rule: if we use the rule to calculate the probability distribution resulting from a sequence of tests, a potentially huge number of conditional probabilities (or probability distributions) is required. This difficulty, coupled with that of extracting probability information from humans, has led to several alternative "rules of inference" for computing likelihood information based on test outcomes (e.g., Shortliffe, 1974). This is an important current research topic.

But there are also ways in which decision theory adds considerable power:

Convenient representation. Utility and probability models are often convenient ways of representing parameters of a problem; they thereby ease parameter modification by a designer or by a user with a slightly different problem. For

example, if a vision operator is modified to use a faster algorithm and therefore less computer time, a small modification to the utility model will suffice to alter the performance of an entire vision system correctly. It would be less obvious how to modify a set of symbolic heuristics that governs the application of the operator.

A simple utility function may express the tradeoffs among the various resources the system consumes (money, elapsed time, etc.). The information that governs the tradeoffs the system actually makes is thus localized and easily modifiable. Some such modifications can be made by the system itself in reaction to complaints about its behavior; the changes could require only simple numerical calculations to compute new parameters for the utility model. It is less obvious how a program should itself "learn" heuristics.

Finally, because decision theory is continually being applied to real-world problems, new models are built, refined, and used. For example, efforts are underway to provide doctors with decision-theory models to help plan the diagnosis and treatment of various diseases (Ginsberg, 1969; Pauker & Kassirer, 1975). Computer aids to such decision making can take advantage of the models.

Ubiquity of planning. Such models are not limited to application in traditional "AI" domains. For example, an optimizing compiler embarks upon substantial symbolic reasoning to plan efficient object code for a program; sophisticated optimizers measure or estimate how often a section of code is executed and use this as an estimate of the utility of an optimization. An extended utility structure would permit trading off different forms of optimization and including the user's utility function. Automatic programming, and in particular automatic coding (Low, 1974), seem to involve the same kinds of planning and elaboration mechanisms presented here.

Optimal planning. A decision–theoretic model of a planning process itself can be used to make planning decisions and thus to control allocation of effort to planning tasks. Many AI programs such as planners, problem solvers, parsers, and "understanders" require such guidance in the application of available methods: Is it more important to plan further ahead or to investigate detail of the current plan (Sacerdoti, 1974)? How far should consequences of a situation be investigated (Rieger, 1975)? Increasingly, this problem becomes one of controlling a number of processes which are "triggered" by various changes in the world model, and which are responsible for exploring consequences of the change (Bobrow & Winograd, 1976). If two alternative parsings of a sentence appear similar in a crude analysis, should one be examined in detail, or should both be explored uniformly (Paxton & Robinson, 1973)? How are alternative hypotheses pursued (Woods, 1974)?

Even if the plans themselves have constant utility, optimal planning is useful. For example, in a theorem prover, we are given a set of clauses and must decide which of several resolutions to make; if we can calculate the cost of planning a solution from a given set of clauses, we choose the resolution which gives rise to

the lowest planning cost. Thus although the space of outcome utilities is constant, the utilities of various alternative planning approaches are not. This second space has been important to the development of search programs; it corresponds, for example, to the evaluation functions in game-playing programs.

When the costs, uncertainties, and outcomes of the planning process itself are considered in controlling a planning and execution system, the system does "optimal planning." Although the plans generated may not be optimal, the entire process, including planning, is optimal. This suggests an extended notion of admissibility that includes consideration of planning costs.

Detection problems. AI has embraced a number of problems that have significant *detection* components: speech understanding and machine vision are the most obvious examples. The problems of efficient detection, and especially of uncertainty in the results, are at the heart of decision theory. In an AI setting, the knowledge gained from detection operations must be incorporated into higher-level reasoning that has significant symbolic components. It is perhaps in these problems that the approach we propose is most advantageous, for it unifies inherently numerical computation (detection) with symbolic reasoning (understanding). Indeed, it is these areas that gave rise to the approach and saw early applications (Bolles, 1976; Garvey & Tenenbaum, 1974; Tenenbaum, 1973; Yakimovsky & Feldman, 1974).

The fields of AI and decision theory clearly have much to offer each other; each provides insights and techniques for solutions to information-processing problems. Researchers in each discipline should learn from the other.

ACKNOWLEDGMENTS

The authors are grateful to many people, especially Dan Bobrow, whose suggestions greatly improved this paper.

APPENDIX: CLOSED FORMS, LEARNING, AND SENSITIVITY ANALYSIS

If a planner is repeatedly given similar problems to solve, much of the searching and elaboration performed each time is wasteful. Ideally, a strategy could be labeled with a set of conditions under which it is optimal; the optimal strategy can be later retrieved by examining the necessary conditions. Such conditions might take the form of rules. For example, a decision rule for the monkey-and-bananas problem with one box is:

$$\begin{array}{ll} \text{if } d_m < 50 \text{ then} & (\text{if } d_b < 3 \text{ then } A\alpha \text{ else } A\beta) \\ \text{else} & (\text{if } d_b < 8 - d_m/10 \text{ then } A\alpha \text{ else } B\alpha) \end{array}$$

The variables in the rule are d_m, the distance from the monkey to the box, and d_b, the distance from the box to the bananas; all other parameters of the problem are assumed fixed to values of Table 3. The rule does not attempt to compare the

eating strategies to plans for pursuing other goals. Rather, it is a convenient way to retrieve a good strategy based on a small number of symbolic requirements (existence of bananas and box) and some parameters (*C*s, *p*s, and *d*s). After retrieving the best eating strategy, its utility can be compared with that of other plans.

Unfortunately, generating concise rules to cover a wide variety of situations is not a trivial task. We can, of course, always resort to planning and searching decision trees if a precomputed strategy is lacking. Furthermore, the results of each search could be stored for ready reference in the future. But a deeper problem makes this difficult: if small changes to any parameter result in different strategies, the number of rules could grow unreasonably large.

We could try to partition the space of parameter values, associating a rule with each cell of the partition. The problem then is to find an appropriate partition. One technique for developing a partition might be called "learning." Initially, a coarse partition is chosen, and strategies or rules are associated with each cell as needed. In addition, a number is kept with each cell that records the average utility actually achieved by the strategy in the past. If this number falls substantially below the expected utility of the strategy, we suspect that the strategy is not valid throughout the cell, and refine the partition. Such a scheme is used in Yakimovsky and Feldman (1974) under manual control.

Alternatively, sensitivity analysis can help devise a partition. Such analysis, if it can be carried out, can answer questions such as "Over what range of parameter values is a strategy rule such as the example valid?" Insignificant changes in the utility, to which the planner is indifferent, need not cause repartitioning. In simple cases when the symbolic expressions for utility calculations are available, partial differentiation can help determine the effect of parameter changes on the utility.

Sensitivity analysis is useful to the planner even if we are not computing a partition. During elaboration, the usefulness of a test that measures a particular parameter is gauged by the change in utility as a result of a change in parameter value. Also, since many of the parameter values may be only approximate, or even subject to gross errors because of unreliable measurements of the state of nature, the sensitivity analysis can warn of gross changes in strategy within the range of parameter variation.

REFERENCES

Bobrow, D. G., & Collins, A. (Eds.) *Representation and understanding*. New York: Academic Press, 1975.

Bobrow, D. G., & Raphael, B. New programming languages for AI research. *Computing Surveys*, 1974, **6**, No. 3.

Bobrow, D. G., & Winograd, T. An overview of KRL, a knowledge representation language. Computer Sciences Laboratory, Xerox Palo Alto Research Center, 1976. Published in *Cognitive Science* **1**, 1977, 3–46.

Bolles, R. Vision for automated assembly. PhD thesis, Computer Science Dept., Stanford University, 1976.

Chernoff, H., & Moses, L. *Elementary decision theory.* New York: Wiley, 1959.

Cozzolino, J. M. Search for an unknown number of objects of nonuniform size. *Operations Research,* 1972, **20,** 293.

deGroot, M. H. *Optimal statistical decisions.* New York: McGraw-Hill, 1970.

Fahlman, S. A planning system for robot construction tasks. *Artificial Intelligence,* 1974, **5,** No. 1.

Feldman, J. A., Low, J. R., Taylor, R., and Swinehart, B. Recent developments in SAIL—An ALGOL-based language for artificial intelligence. *Proceedings of the Fall Joint Computer Conference,* 1972.

Fikes, R. E. Knowledge representation in automatic planning systems. Tech. Note 119, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California, 1976.

Fikes, R. E., & Nilsson, N. J. STRIPS: A new approach to the application of theorem proving in problem solving. *Artificial Intelligence,* 1971, **2,** 189.

Garvey, T., & Tenenbaum, J. M. On the automatic generation of programs for locating objects in office scenes. *Proceedings of the Second International Joint Conference on Pattern Recognition,* IEEE 74-CH0885-4C, August 1974, p. 162.

Ginsberg, A. S. Decision analysis in clinical patient management with an application to the pleural effusion problem. PhD thesis, Stanford University, 1969.

Grape, G. Model based (intermediate level) computer vision. PhD thesis, Computer Science Dept., Stanford University, Stanford, California, 1973.

Hart, P., Nilsson, N. J., & Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics,* 1968, **SSC-4,** 100.

Lawler, E. and Wood, D. Branch and bound methods: a survey. *Operations Research,* 1966, **14,** 699.

Low, J. R. Automatic coding: choice of data structures. Stanford Artificial Intelligence Project AIM 242, Stanford, Calif., August 1974.

Matheson, J. E. The economic value of analysis and computation. *IEEE Transactions on Systems Science and Cybernetics,* 1968, **SSC-4,** 325.

Munson, J. H. Robot planning, execution and monitoring in an uncertain environment. *Proceedings of the Second International Joint Conference on AI,* September, 1971, p. 338.

Newell, A., Shaw, J., & Simon, H. Chess playing programs and the problem of complexity. *IBM Journal of Research and Development,* 1958, **2,** 320. Reprinted in E. Feigenbaum & J. Feldman (Eds.), *Computers and thought.* New York: McGraw-Hill, 1963.

Nilsson, N. J. Artificial intelligence. *Proc. IFIP Congress,* 1974, p. 778.

Nilsson, N. J. *Problem-solving methods in artificial intelligence.* New York: McGraw-Hill, 1971.

Pauker, S. G., & Kassirer, J. P. Therapeutic decision making: A cost-benefit analysis, *New England Journal of Medicine,* 1975, **239,** 229–234.

Paxton, W. H., & Robinson, A. E. A parser for a speech understanding system. *Proceedings of the Third International Joint Conference on AI,* August, 1973, p. 216.

Pohl, I. The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computational issues in heuristic problem solving. *Proceedings of the Third International Joint Conference on AI,* August 1973, p. 12.

Raiffa, H. *Decision analysis, introductory lectures on choices under uncertainty.* Reading, Mass.: Addison Wesley, 1970.

Rieger, C. J. Conceptual memory. In R. Schank (Ed.), *Conceptual information processing.* Amsterdam: North-Holland, 1975.

Rulifson, J. F., Derksen, J. A., & Waldinger, R. J. QA4: A procedural calculus for intuitive reasoning. Tech. Note 73, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California, 1972.

Sacerdoti, E. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence,* 1974, **5,** 115–135.

Sacerdoti, E. A structure for plans and behavior. Tech. Note 109, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California, 1975.

Samuel, A. L. Some studies in machine learning using the game of checkers, II. Recent progress. *IBM Journal of Research and Development*, 1967, **11**, No. 6.

Shortliffe, E. H. MYCIN: A rule-based computer program for advising physicians regarding antimicrobial therapy selection. AI Memo 251; Stanford Artificial Intelligence Project, Stanford, Calif., October, 1974.

Slagle, J. R., & Lee, R. C. T. Application of game tree searching techniques to sequential pattern recognition. *Communications of the ACM*, 1971, **14**, 103.

Sproull, R. F. Strategy construction using a synthesis of heuristic and decision-theoretic methods. PhD thesis, Computer Science Dept., Stanford University, 1977.

Taylor, R. H. Assembly Robot Program Automation. PhD thesis, Computer Science Dept., Stanford University, 1976.

Tenenbaum, J. M. On locating objects by their distinguishing features in multisensory images. *Computer graphics and image processing*, 1973, **2**, No. 3/4.

Tversky, A., & Kahneman, D. Judgement under uncertainty: heuristics and biases. *Science*, 1974, **185**, No. 4157, 1124.

Woods, W. A. Motivation and overview of BBN Speechlis: An experimental prototype for speech understanding research. *IEEE Symposium on Speech Recognition*, IEEE 74-CH0878-9 AE, April 1974.

Yakimovsky, Y., & Feldman, J. Decision theory and artificial intelligence: I. A semantics-based region analyzer. *Artificial Intelligence*, 1974, **5**, 349–371.