INFORMATION AND CONTROL 40, 1–19 (1979)

On the Complexity of Decision Trees, the Quasi-Optimizer, and the Power of Heuristic Rules

NICHOLAS V. FINDLER

Department of Computer Science, State University of New York at Buffalo, 4226 Ridge Lea Road, Amherst, New York 14226

AND

JAN VAN LEEUWEN

Department of Computer Science, University of Utrecht, 3508-TA Utrecht, The Netherlands

The power of certain heuristic rules is indicated by the relative reduction in the complexity of computations carried out, due to the use of the heuristics. A concept of complexity is needed to evaluate the performance of programs as they operate with a varying set of heuristic rules in use. We present such a complexity measure, which we have found useful for the quantitative evaluation of strategies in the context of a long-term research project on poker. We define our measure in terms of the level-complexity for decision trees and study the measure for the relevant class of decision trees with a fixed but arbitrary number of levels, h, and k leaves (all at the last level). We determine the best and worst case distributions for the levels in this measure. We show, for instance, that for the smallest value, $L_h(k)$, and the largest value, $U_h(k)$, which the level-complexity for such trees can have, $\lim_{k\to\infty} L_h(k)/\log^h(k) = 1$ and $\lim_{k\to\infty} U_h(k)/\log(k) = 1$. We show also that the level-entropy assumes its maximum value of $\log(k)$ just when the path-entropy reaches its minimum value over all trees with k leaves but, in general, the values of either measure can vary in a rather unrelated manner. We give a detailed example of how the complexity measure is used in evaluating the power of heuristic rules used in assessing the performance of the Quasi-Optimizer (QO), a program currently under development. The objectives of the QO are explained, and the three main phases of operation of the program are described within the framework of the poker project.

1. INTRODUCTION

Decision trees are a convenient information structure for describing both deterministic and stochastic, multistage decision processes (see, e.g., Burge, 1958; Jardine and Sibson, 1971; Salton, 1968; and Picard, 1965). Static strategies fit this representation naturally, but we show that dynamically evolving, learning strategies can also be characterized in this form.

0019-9958/79/010001-19$02.00/0

Discrimination trees, which grow automatically as demanded by the task at hand, have been used to simulate various cognitive processes (see, e.g., Feigenbaum, 1961; Simon and Feigenbaum, 1964; Simon and Gilmartin, 1973). In a competitive environment, different strategies interact and affect each other. For example, human and machine players are pitted against each other in a complex large-scale program of draw poker, which Findler and his co-workers have used over the past few years in their studies on decision making under uncertainty and risk (Findler, 1973, 1977; Findler *et al.*, 1972; Findler *et al.*, 1973; Findler *et al.*, 1974). Here, the short-term and long-term goals of *some* of the machine players are expressed in the form of decision trees.

It is not relevant for our present purposes to discuss the objectives, the methodology, or the results of the above work. However, we outline in Section 3 the conceptual framework of and the technical approach used in one of the subprojects, the Quasi-Optimizer (QO), which is under development. This program would generate a normative theory (a quasi-optimum strategy) out of descriptive theories (the strategies used by individual players). In constructing the quasi-optimum strategy, the program makes use of certain heuristic rules which reduce the domain of search for some decision processes. In attempting to measure the effectiveness and power of the heuristics employed (cf. Simon and Kadane, 1976), we were led to define a natural and plausible complexity measure for decision trees. We next investigate the theoretical limits on the complexity of decision trees, and show how the concept has practical relevance for the evaluation of our heuristic rules.

2. The Level-Complexity of Decision Trees

In the poker-playing application that motivated the results of this section, a program called the Quasi-Optimizer combines according to certain heuristic rules several trees representing individual game-playing strategies into a tree representing a composite strategy. Since the use of different heuristics may result in different composite trees, it is useful to measure the "power" of such heuristics by comparing the complexity of the input trees with that of composite tree.

Let us assume that all trees are of equal height h (every decision criterion is considered by each input strategy; if necessary, by a nonbranching node). Let the number of nodes in the jth level of a tree T be K_j (for $0 \leqslant j \leqslant h$), and let the outdegree of the kth node from the left in this level be r_{jk}. Logarithms are always taken in base 2.

The average number of comparisons at the kth node in level j is adequately measured by $\log r_{jk}$. Assuming that all nodes at level j have equal probability of being reached, it follows that at this level the program makes an average number of comparisons equal to

$$C_j = K_j^{-1} \cdot \sum_{k=1}^{K_j} \log r_{jk} \tag{1}$$

before it moves on to the $(j + 1)$st level. If a specific probability distribution of the reachability of nodes (of the values of decision criteria) is known, then (1) becomes

$$C_j' = \sum_{k=1}^{K_j} p_{jk} \cdot \log r_{jk}\,, \tag{1'}$$

where p_{jk} denotes the probability that the game situation at the jth level in T is driven through the kth node from the left, and obviously

$$\sum_{k=1}^{K_j} p_{jk} = 1. \tag{1''}$$

We do not pursue that refinement here, and only consider the simplified uncertainty situations covered by (1). In poker, the particular probabilities are obtainable either mathematically (for example, the probability distribution of possible hands) or from experience (for example, the type of games played against a given set of opponents). One could in fact approximate the probabilities to obtain various models, each with its own measure. We do not do so here and the measure of complexity is analyzed only when all p_{jk} are equal K_j^{-1}.

The complexity of a decision tree may now be defined as the total number of comparisons par level a player makes on the average in progressing from the root downward to the leaf level.

DEFINITION. The level-complexity of the tree T is

$$C_T = \sum_{j=0}^{h-1} C_j\,. \tag{2}$$

The concept of complexity is related to but is different from the concept of path-entropy introduced by Green (1973). One might call the quantity expressed in (2) the "level-entropy," and we explore below how it is substantially different from the path-entropy concept and in what sense it measures an entirely different distribution of decisions in a tree (see Section 2.3).

We first explore the (asymptotic) lower and upper bounds of the level-complexity of decision trees, and investigate the theoretical limits of the measure.

2.1. *A Lower Bound for Level-Complexity*

We have to introduce a few more conventions before we can formally treat the level-complexity of a decision tree T.

The decision trees we consider always have h levels ($h \geqslant 1$), with all leaves occurring in the hth ("the last") level (see Fig. 1).

Such a tree may be characterized in part by the number of nodes in the various consecutive levels

$$1 = K_0 \leqslant K_1 \leqslant \cdots \leqslant K_h .$$

The generic notation for any tree with these level-characteristics is

$$T[K_0 , K_1 ,..., K_h].$$

It depends on the precise distribution of edges between the levels (thus on the r_{jk}) what the complexity of T is.

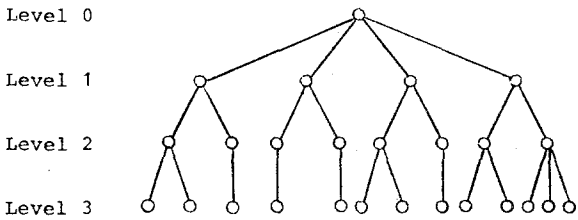

FIG. 1. A decision tree with three levels ($h = 3$).

At any level j with $K_j = K_{j+1}$, we must have $r_{jk} = 1$ for $1 \leqslant k \leqslant K_j$, and it easily follows that such levels make a contribution of zero to the level-complexity of the tree. To get a meaningful analysis, we assume that $K_j \neq K_{j+1}$ for all j throughout this section.

DEFINITION. $L_h(K) = \min\{C_T \mid T \text{ is a } T[K_0 ,..., K_h] \text{ with } K_0 = 1, K_h = K, \text{ and arbitrary } K_1 ,..., K_{h-1} \text{ with } K_0 < K_1 < \cdots < K_h\}$.

Consider any such decision tree T. Level 0 always contributes $\log K_1$. We determine how to achieve the smallest possible contribution of the jth level ($j > 0$), which solely depends on the distribution of the r_{jk}'s.

LEMMA 2.1.1. *The minimum level-complexity of a decision tree with level-characteristics* $K_0 < K_1 < \cdots < K_h$ *is*

$$\log K_1 + \sum_{j=1}^{h-1} K_j^{-1} \cdot \log(K_{j+1} - K_j + 1). \tag{3}$$

Proof. The jth level contributes $K_j^{-1} \cdot \log r_{j1} \cdots r_{jK_j}$. The product of r's must be minimized under the assumptions: $r_{jp} \geqslant 1$ and $r_{j1} + \cdots + r_{jK_j} = K_{j+1}$.

The minimum is achieved when all r_{jp}'s are 1, except for one which must be

$K_{j+1} - K_j + 1$. To show this, let us assume that the r_{jp}'s were chosen so as to have a minimal product. Let $1 \leqslant l \leqslant K_j$ be such that $r_{jl} = \max\{r_{jp} \mid 1 \leqslant p \leqslant K_j\}$. If $r_{jl} = K_{j+1} - K_j + 1$, we are done. Assume that $r_{jl} < K_{j+1} - K_j + 1$. There must be a $p \neq l$, $1 \leqslant p \leqslant K_j$ such that $r_{jp} \geqslant 2$ (whereas $r_{jp} \leqslant r_{jl}$). Consider how the product of r's changes if we let r_{jp} be $r_{jp} - 1$ and r_{jl} be $r_{jl} + 1$ (which keeps the sum of r's the same)

$$(r_{jp} - 1)(r_{jl} + 1) = r_{jp}r_{jl} - (r_{jl} - r_{jp}) - 1 < r_{jp} \cdot r_{jl}\,.$$

The product of r's decreases, contradicting that it was choosen to be minimal.

The minimum contribution of the jth level is $K_j^{-1} \cdot \log(K_{j+1} - K_j + 1)$. ∎

Obviously $L_h(K) = \log K$ for $h = 1$. For $h \geqslant 2$, one can have decision trees with K leaves of a much smaller level-complexity because the branching behavior is averaged over more levels. It cannot be arbitrarily small for a fixed h, though.

Let $\log^{(i)}(x)$ denote the ith iterate of $\log(x)$

$$\log^{(i)}(x) = \underbrace{\log \cdots \log}_{i}(x) \tag{4}$$

LEMMA 2.1.2. *One can construct a $T[K_0, \ldots, K_h]$ tree with $K_h = K$ such that $C_T = \log^{(h)}(K) + \mathcal{O}(h)$ for $K \to \infty$.*

Proof. We only need to consider K values large enough such that $\log(K)$, $\log^{(2)}(K), \ldots, \log^{(h)}(K)$ have strictly decreasing integral parts.

Consider any decision tree with level-characteristics

$$K_j = [\log^{(h-j)}(K)]$$

for $1 \leqslant j \leqslant h$ (with $K_0 = 1$ and defining $\log^{(0)} K = K$). By Lemma 2.1.1, we may arrange the tree such that the jth level ($1 \leqslant j \leqslant h$) contributes a minimum complexity of

$$\begin{aligned} K_j^{-1} \cdot \log(K_{j+1} - K_j + 1) &\leqslant K_j^{-1} \cdot \log(K_{j+1}) \\ &\leqslant [\log^{(h-j)}(K)]^{-1} \cdot \log \log^{(h-j-1)}(K) \\ &= [\log^{(h-j)}(K)]^{-1} \cdot \log^{(h-j)}(K) \\ &= \mathcal{O}(1) \qquad \text{for} \quad K \to \infty. \end{aligned}$$

The total level-complexity of the tree is

$$\log K_1 + \underbrace{\mathcal{O}(1) + \cdots + \mathcal{O}(1)}_{(h-1)\text{times}} = \log^{(h)}(K) + \mathcal{O}(h)$$

for $K \to \infty$. ∎

There are numerous variants of the construction in Lemma 2.1.2, but we prove that none of these alternative constructions can succeed in finding a decision tree of complexity substantially less than $\log^{(h)}(K)$ as $K \to \infty$. It means that we found the order of magnitude for the minimum level-complexity for fixed h, as $K \to \infty$.

THEOREM 2.1.3. $\lim_{K\to\infty} L_h(K)/\log^{(h)}(K) = 1$.

Proof. The result is trivial for $h = 1$. Assume $h \geqslant 2$. By Lemma 2.1.2 we know that $L_h(K) \leqslant \log^{(h)}(K) + \mathcal{O}(h)$ for $K \to \infty$. The result follows once we show that for no ϵ, $0 < \epsilon < 1$, there can be an infinite sequence of K-values for which $L_h(K) \leqslant \epsilon \cdot \log^{(h)}(K)$.

Suppose by way of contradiction that there was an ϵ, $0 < \epsilon < 1$, such that $L_h(K) \leqslant \epsilon \cdot \log^{(h)}(K)$ for infinitely many K-values. It follows that, for each such K-value, we can find a decision tree with level-characteristics $K_0 < \cdots < K_h$ (and $K_h = K$) such that

$$\log(K_{j+1} - K_j + 1) \leqslant K_j \cdot \epsilon \cdot \log^{(h)}(K) \tag{5}$$

for $1 \leqslant j < h$. Let $\delta(K) = 1/[4\epsilon \cdot \log^{(h)}(K)]$. We prove by induction, for j from h to 0, that for all K large enough

$$K_j \geqslant \delta(K) \cdot \log^{(h-j)}(K). \tag{6}$$

It is obviously true for $j = h$ (choose K so large that, for instance, $\log^{(h)}(K) \geqslant 1/4\epsilon$). Suppose (6) holds for $j + 1$. We claim that for K large enough

$$K_j \geqslant 2\delta(K) \cdot \log(K_{j+1}) = \log(K_{j+1})/[2\epsilon \cdot \log^{(h)}(K)]. \tag{7}$$

Suppose $K_j < \log(K_{j+1})/[2\epsilon \cdot \log^{(h)}(K)]$. By (5) we would have the following condition on K_{j+1}:

$$K_{j+1} - (K_{j+1})^{1/2} + 1 < \log(K_{j+1})/[2\epsilon \cdot \log^{(h)}(K)].$$

As the induction hypothesis (6) holds for $j + 1$, and K_{j+1} grows bigger and bigger as K gets larger, the inequality is violated for all sufficiently large K values.

Using the induction hypothesis, it follows from (7) that

$$\begin{aligned} K_j &\geqslant \{1/[2\epsilon \cdot \log^{(h)}(K)]\} \cdot \{\log(\log^{h-j-1}(K)) - \log^{(h+1)}(K) - \log 4\epsilon\} \\ &= \{\log^{(h-j)}(K)/[2\epsilon \cdot \log^{(h)}(K)]\} \cdot \{1 - (\log^{(h+1)}(K) + \log 4\epsilon)/\log^{(h-j)}(K)\}. \end{aligned} \tag{8}$$

By choosing K sufficiently large, we conclude (6) again.

It follows that (6) holds for all $1 \leqslant j \leqslant h$. The lower bounds on the K_j imply

a lower bound on the level-complexity of any tree with these characteristics. Let us consider just the contribution of level 1. It is

$$\begin{aligned} \log K_1 &\geqslant \log(\delta(K) \log^{(h-1)}(K)) \\ &= \log^{(h)}(K) - \log^{(h+1)}(K) - \log 4\epsilon \\ &= \log^{(h)}(K) \cdot \left\{1 - \frac{\log^{(h+1)}(K) + \log 4\epsilon}{\log^{(h)}(K)}\right\}. \end{aligned} \tag{9}$$

It is easily seen that the factor of $\log^{(h)}(K)$ approaches 1 as $K \to \infty$ and the total complexity must outgrow $\epsilon \cdot \log^{(h)}(K)$ for all sufficiently large K, contrary to our assumption. ∎

2.2 *An Upper Bound for Level-Complexity*

We now establish a best possible upper bound on the level-complexity of decision trees with h levels.

DEFINITION. $U_h(K) = \max\{C_T \mid T \text{ is a } T[K_0, \ldots, K_h] \text{ with } K_0 = 1, K_h = K, \text{ and arbitrary } K_1, \ldots, K_{h-1} \text{ with } K_0 \leqslant \cdots \leqslant K_h\}$. (Compared to the definition of $L_h(K)$, it is not necessary this time to demand that $K_{j+1} > K_j$ for all j.)

The first level of a decision tree always contributes $\log(K_1)$. We determine how to distribute the r_{jp} values in the jth so as to maximize the complexity of this level.

We note that the level-complexity of a decision tree is minimal if in each level as many edges as possible are concentrated in a single node (Lemma 2.1.1). Exactly the opposite appears to be true in case we want to maximize complexity, and we prove that the contribution of each level is maximum when the edges leading to the next level are distributed over the K_j nodes as uniformly as possible.

LEMMA 2.2.1. *The maximum level-complexity of a decision tree with level-characteristics* $K_0 \leqslant \cdots \leqslant K_h$ *is*

$$\log K_1 + \sum_{j=1}^{h-1} \log \left\lfloor \frac{K_{j+1}}{K_j} \right\rfloor^{1-\epsilon_j} \left\lceil \frac{K_{j+1}}{K_j} \right\rceil^{\epsilon_j}, \tag{10}$$

where $\epsilon_j = (K_{j+1} \bmod K_j)/K_j$ *for* $1 \leqslant j < h$.

Proof. As in Lemma 2.1.1, we must now maximize $r_{j1} \cdots r_{jK_j}$ under the conditions that $r_{jp} \geqslant 1$ and $r_{j1} + \cdots + r_{jK_j} = K_{j+1}$.

The maximum is achieved just in case no two r_{jp}'s differ by more than 1. To show this, let us assume that there were p and q ($1 \leqslant p, q \leqslant K_j$ with $p \neq q$) such that $r_{jp} - r_{jq} \geqslant 2$. Consider how the product of the r's changes

if we let r_{jp} be $r_{jp} - 1$ and r_{jq} be $r_{jq} + 1$ (which does keep the sum of the r's the same)

$$(r_{jp} - 1)(r_{jq} + 1) = r_{jp} \cdot r_{jq} + (r_{jp} - r_{jq}) - 1 > r_{jp} \cdot r_{jq} .$$

The product has become larger, contrary to the assumption of maximality.

It remains to be shown that the condition on the r_{jp} indeed uniquely determines the value of $r_{j1} \cdots r_{jK_j}$.

It is easily seen that the r_{jp}'s must be equal to $\lfloor K_{j+1}/K_j \rfloor$ or $\lceil K_{j+1}/K_j \rceil$ (otherwise one could find two r_{jp}'s more than 1 apart). Suppose s_j of the r_{jp}'s are equal to $\lfloor K_{j+1}/K_j \rfloor = K_{j+1}/K_j - \epsilon_j$, and thus $K_{j+1}/K_j = \lceil K_{j+1}/K_j \rceil + 1 - \epsilon_j$. Strictly, it holds only for $\epsilon_j \neq 0$ but the analysis yields the same final result if one takes $\epsilon_j = 0$. Let t_j be such that

$$s_j + t_j = K_j .$$

Then

$$s_j \lfloor K_{j+1}/K_j \rfloor + t_j \lceil K_{j+1}/K_j \rceil = K_{j+1} . \tag{11}$$

It easily follows that

$$\begin{aligned} s_j &= (1 - \epsilon_j)K_j , \\ t_j &= \epsilon_j K_j , \end{aligned} \tag{12}$$

and the complexity contributed by the jth level is equal to

$$\begin{aligned} &K_j^{-1} \cdot \log \lfloor K_{j+1}/K_j \rfloor^{(1-\epsilon_j)K_j} \cdot \lceil K_{j+1}/K_j \rceil^{\epsilon_j K_j} \\ &\quad = \log \lfloor K_{j+1}/K_j \rfloor^{1-\epsilon_j} \cdot \lceil K_{j+1}/K_j \rceil^{\epsilon_j}. \quad \blacksquare \end{aligned}$$

We can now derive the interesting result that the level-complexity of any decision tree (of the type we are considering) is bounded by the log of the number of leaves.

THEOREM 2.2.2. *For any decision tree T with h levels and K leaves we have*

$$C_T \leqslant \log(K)$$

and equality holds if and only if $K_j \mid K_{j+1}$ for $0 \leqslant j < h$ (where $K_0, \ldots, K_h$ are the level-characteristics of T).

Proof. The maximum level-complexity a tree T can have is given in Lemma 2.2.1.

The result is immediate for $h = 1$, so let us assume that $h \geqslant 2$. Consider an arbitrary expression

$$\lfloor K_{j+} / K_j \rfloor^{1-\epsilon_j} \cdot \lceil K_{j+1}/K_j \rceil^{\epsilon_j}.$$

If $K_j \mid K_{j+1}$ then it equals K_{j+1}/K_j.

If $K_j \nmid K_{j+1}$ then $K_j < K_{j+1}$ (strictly) and $0 < \epsilon_j < 1$. Let $l_j = \lfloor K_{j+1}/K_j \rfloor$, $u_j = \lceil K_{j+1}/K_j \rceil$, and observe that now $u_j = l_j + 1$. By the generalized Bernoulli inequality (Mitrinovic, 1964), we have

$$\begin{aligned}(1 + 1/l_j)^{\epsilon_j} &< 1 + \epsilon_j/l_j \\ &\Rightarrow l_j(1 + 1/l_j)^{\epsilon_j} < l_j + \epsilon_j \\ &= l_j^{1-\epsilon_j} \cdot (l_j + 1)^{\epsilon_j} < l_j + \epsilon_j \,. \end{aligned} \tag{13}$$

It follows that $\lfloor K_{j+1}/K_j \rfloor^{1-\epsilon_j} \cdot \lceil K_{j+1}/K_j \rceil^{\epsilon_j} < \lfloor K_{j+1}/K_j \rfloor + \epsilon_j = K_{j+1}/K_j$.

In both cases the level-complexity may be estimated as

$$\leqslant \log K_1 + \log K_2/K_1 + \cdots + \log K_h/K_{h-1} = \log K$$

but equality holds only if $K_j \mid K_{j+1}$ for all relevant j's. ∎

It is now easy to prove that the maximum complexity of decision trees with K leaves must always be "close" to $\log K$ even if we demand that $K_j < K_{j+1}$ for all j.

PROPOSITION 2.2.3. $\lim_{K\to\infty} U_h(K)/\log K = 1$.

Proof. By Theorem 2.2.2, we know that $U_h(K) \leqslant \log K$. For each K this bound is achievable with a decision tree of level characteristics $K_0 = 1, K_1 = K$, and $K_j = K_{j+1}$ for $1 \leqslant j < h$.

If we require that $K_j < K_{j+1}$ for $0 \leqslant j \leqslant h$, then we can get arbitrarily close to $\log K$ (as $K \to \infty$) by taking a decision tree with characteristics $K_j = K - h + j$. ∎

The results of Theorem 2.1.3 and Proposition 2.2.3 completely determine (asympotically) the range of level-complexity for decision trees with h levels and K leaves.

2.3. *A Comparison of Level-Entropy and Path-Entropy*

The concept of level-complexity as we defined it is by no means the only measure for decision trees that has been considered. Whereas we have attempted to find a meaningful global measure, Green (1973) introduced the concept of path-entropy to estimate the average number of comparisons needed to reach a specific leaf. In this section, we argue that level-complexity and path-entropy behave quite differently and measure entirely different kinds of distributions in a tree.

For any leaf l in a tree T, let $\pi_l = s_1, \ldots, s_m$ be the unique path from the root to l. A measure for the number of comparisons needed to reach l is

$$H(\pi_l) = \sum_{i=1}^{m} \log \text{degree}(s_i).$$

Let T contain K leaves $l_1, \ldots, l_K$.

DEFINITION. The path-entropy of T is $H(T) = (1/K) \sum_{i=1}^{K} H(\pi_{l_i})$.

It is interesting to contrast our results for level-complexity with those of Green for path-entropy. For one thing, Green showed that

$$\log K \leqslant H(T) \leqslant \frac{(K-1)(K+2)}{2} \tag{14}$$

for any tree T with K leaves. This is completely disjoint from the range for level-complexity, which we established for our trees. In fact, it appears that the level-complexity of a tree is maximal (see Theorem 2.2.2) just when the path-entropy is the smallest! It is easy to see that

$$\text{level-complexity} \leqslant \text{path-entropy} \tag{15}$$

for all trees, with equality holding only if both entities assume the value of $\log K$.

A "large" value of level-complexity was earlier an indication of a rather even distribution of effort at each level, which seems to concur with the fact that for such trees the path-entropy tends to be "small." If an even distribution is desired, the effectiveness of the QO may be measured by the increase of level-complexity towards $\log K$. It is not true, though, that a rising value of the level-complexity through local reorganization of a tree means a falling value of the path-entropy (or conversely).

PROPOSITION 2.3.1. $C_{T_1} < C_{T_2} \not\Rightarrow H(T_1) > H(T_2)$

Proof.

Consider the tree T_1 in Fig. 2, and calculate the change of measure if we move node z to a new father y (in T_2).

Calculating how the contributions in the formulas change, we obtain

$$\begin{aligned} C_{T_1} - C_{T_2} &= \frac{1}{K_1} [\log a + \log b - \log(a-1) - \log(b+1)] \\ &= \frac{1}{K_1} \log \frac{ab}{ab + a - b - 1} \end{aligned}$$

Thus,

(i) the level-complexity increases if $a > b + 1$,

(ii) the level-complexity decreases if $a < b + 1$.

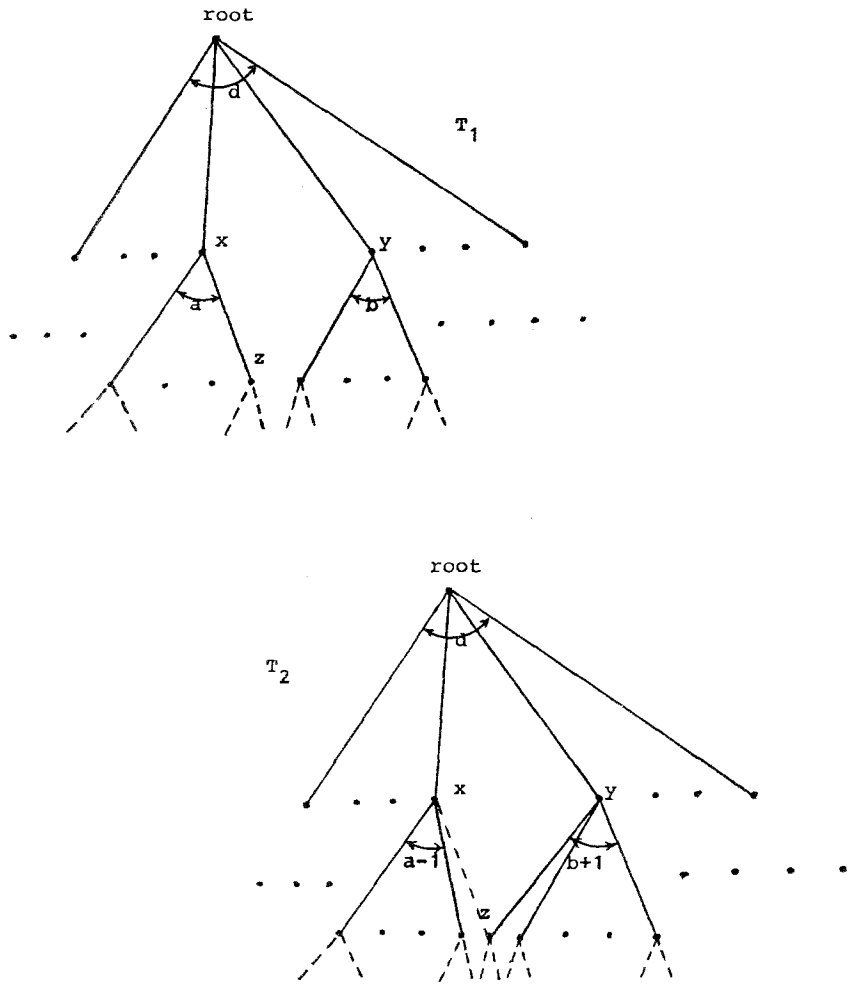

FIG. 2. A change in the tree results in a change in the complexity measure.

The change in path-entropy is harder to compute. Let us identify x, y, and z with their weights, i.e., the number of nodes (in tree T_1) in the subtree of each of these nodes. Then

$$H(T_1) - H(T_2)$$

$$= \frac{1}{K}\left[x \log a + y \log b - (x - z) \log(a - 1) - (y + z) \log(b + 1)\right]$$

$$= \frac{1}{K}\left[x \log \frac{a}{a-1} - y \log\left(1 + \frac{1}{b}\right) - z \log \frac{b+1}{a-1}\right],$$

where we have to assume (obviously) that $x > z$. The freedom in choosing the parameters gives us the desired conclusion:

(i) Assume $a > b + 1$, thus the level-complexity increases. Fixing z, it depends on the relative values of x and y whether $H(T_1) > H(T_2)$ or $H(T_1) < H(T_2)$

(ii) Assume $a < b + 1$, thus $C_{T_1} > C_{T_2}$. Again, a similar conclusion follows. ▮

The proof of Proposition 2.3.1 shows that level-complexity and path-entropy can differ for quite unrelated reasons. As we are not directly interested in the complexity of reaching a leaf but rather want to measure the distribution of decisions throughout the tree, level-complexity is the more advantageous criterion for our present purposes.

3. The Quasi-Optimizer

We discuss the QO program in a general context first. Consider an environment in which several organizations compete to achieve some identical goal. We may assume, for the sake of generality, that a *goal vector* is specified whose components need not be orthogonal in real-life situations. Each organization perceives the task environment by observing and measuring certain relevant parameters. Each organization has a strategy for interpreting its observations, resulting in a course of action leading to goal achievement. At any moment in time, the rules of the competition and the past and current actions of the participants determine the next state of the environment.

One must bear in mind that the environment as perceived by each organization is unclear, because some information may be unavailable, missing (risky or uncertain—whether or not the relevant probability distributions are known, respectively), or obscured by noise. The latter may be caused by latent environmental factors or deliberate obfuscation by competitors. Further, conflicts and biases within the organizations can also lead to erroneous measurements. If the decisions based on such information are less correct than those of the competitors, resources will be wasted and goal attainment will be further removed.

Let us now consider the situation where a new organization wants to enter the competition. In trying to construct a strategy for itself, it observes the environment and the actions by other organizations. It will, in fact, build models of their strategies and extract the components which appear to be the best under particular conditions. An appropriate combination of such components should result in a "quasi-optimum" strategy which is normative against the given set of competitors.

We now describe the major phases of the QO program. Although its goals can

be described in the above general context, we use the conceptual framework of poker for the sake of the following discussion [see Findler (1977) for details of the project].

3.1 *Gaming Experiments with Input Strategies*

Input strategies of competing players are provided in terms of impenetrable "black box" programs. In order to analyze and process the information embedded in them, one must represent the inferred strategies in the form of some uniform structure. The two obvious choices are Production Systems [see, e.g., Waterman (1970)] and Decision Trees. In view of our experience with the latter in the poker project, we have chosen decision trees. Decision trees are relatively easy to to generate, compare, and modify, both in terms of structure and contents.

The QO program is first given the set of all possible decision criteria (levels of discrimination on the trees) and the maximum range of each criterion. The values of the decision criteria can be numerical (e.g., "the number of players still alive is three"), in the form of rank numbers (e.g., "the player who has just raised is the most easy to outbluff"), or symbolic (e.g., "the games played so far are on the conservative side"). The objective of this phase of the QO is to construct decision trees which characterize the strategy of every machine player through a simulated question–answer technique ("what would you do if the game-situational and opponent-personality variables assume such-and-such value combinations?"). The information initially given about all decision criteria and their ranges makes sure that the experiments span the whole game space. Whenever some variable has no effect of discrimination in a strategy, the program inserts nonbranching nodes at the appropriate level. Thus every decision tree will be of equal height.

3.2 *The Construction of a Super Tree*

The next phase of the QO program aims at building a "Super Tree", which carries all the necessary information contained in the individual decision trees. The output of phase one, the descriptive theories of player strategies, provides the input for this second phase. The Super Tree must be as little redundant as possible so that the final, optimization phase may be within manageable limits.

A crude version of the Super Tree would be simply the "join" of the input trees (see later). We need some heuristic rules to eliminate much of its redundancy, provided the rules are computationally inexpensive and effective. We describe the adopted heuristic in detail in the next section. After optimizing in a top-down manner, the QO must finally reach the leaf level. Specific game actions, as the result of discrimination decisions, are attached to the leaf nodes of the Super Tree. As different strategies may and often do recommend different

actions, some leaves may be populated by several conflicting actions. It is left to the third phase of the QO program to eliminate this inconsistency.

The third phase has two objectives: (a) selection of the best of possibly several decisions at the leaf level, and (b) further simplification of the Super Tree by possibly merging levels of decision criteria. The latter could be accomplished by chunking some variables together. We are not concerned with such a process in the present paper.

4. The Power of the Heuristics Used for Simplifying the Crude Super Tree and a Numerical Example

The QO program explained in the previous section takes as input a collection of decision trees of varying complexities, and forms a crude Super Tree as the "join" of these trees. It then attempts to optimize the tree by some heuristic rules. We pick up the actions of the QO as it forms the Super Tree. Consider the three input trees in the top part of Fig. 3 as an example.

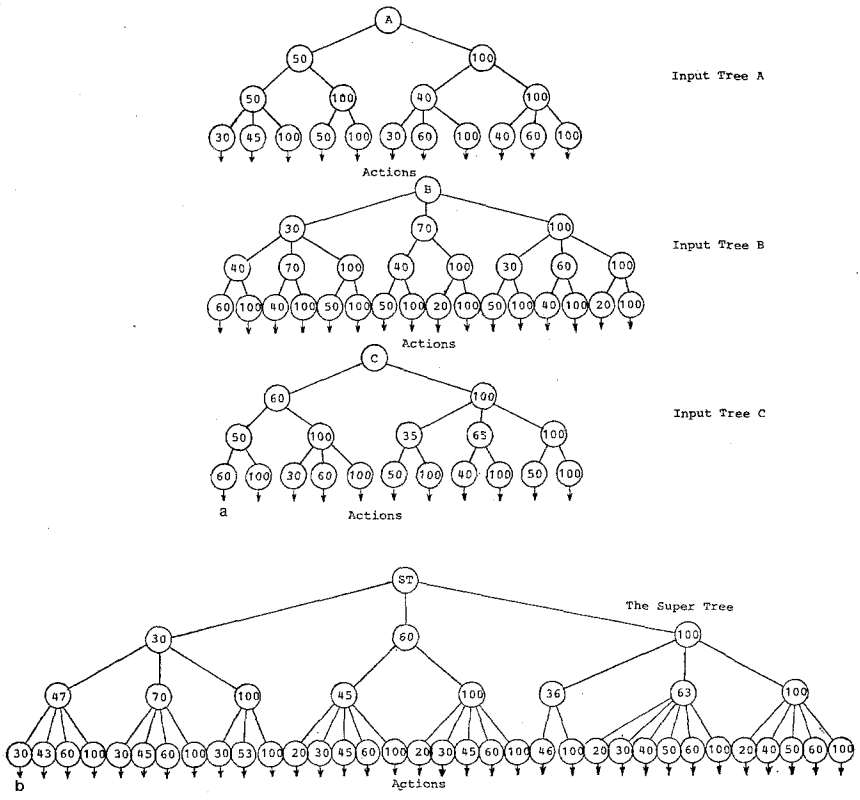

FIG. 3. The construction of the Super Tree from three input trees.

The crude version of the Super Tree is the "join" of the input trees. It means that the number of branches emanating from a node of the crude Super Tree is the algebraic sum of the branches (discounting identical subranges) emanating from the equivalent nodes of every input tree. Figure 4 illustrates this idea by showing the first level of a crude Super Tree computed from the levels one of the three input trees. The result seems to contain not only all the necessary information but much more. We now describe in detail the adopted heuristic for eliminating much of the redundancy.

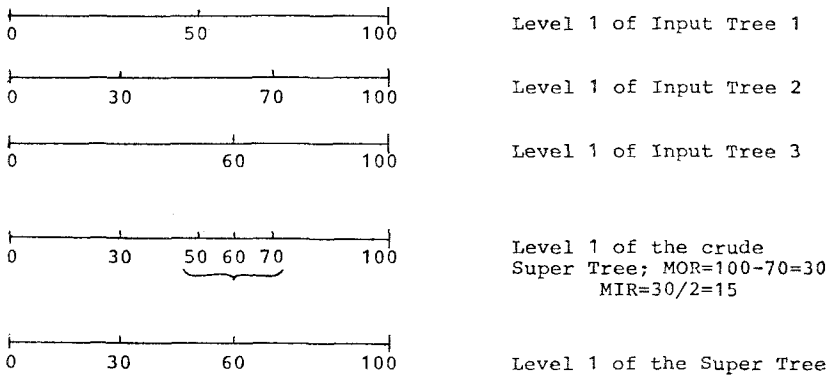

FIG. 4. Computations leading to the level 1 values of the Super Tree.

Let us first convert rank numbers and values of symbolic discrimination criteria to numerical ranges. (This is no problem as they were obtained from numerical comparisons in the first place). Next we can define for each level a Maximum Input Resolution (MIR). It is the shortest distance on any input tree between any two adjacent demarkation points (DPs)—that is, the smallest subrange which belongs to any branch coming to this level. (This is 30 from Input Tree B on Fig. 1). The Maximum Output Resolution (MOR)—that is, the shortest distance between any two adjacent DPs at this level of the Super Tree—is taken to be one-half of MIR (15 in our example on Fig. 4). There is no need to use a more refined mode of discrimination. Our reason for halving the MIR is that DPs are fuzzy because (i) the strategies of players are fuzzy, (ii) the models themselves are error-prone and not completely consistent. We have assumed that the extent of fuzziness of a DP on the input trees is one-quarter of the adjacent subranges, pointing both ways into the subranges (see Fig. 5). What is left "certain," one-half of MIR, is chosen to be MOR.

The program now locates groups of DPs at level one of the crude Super Tree so that within each group, adjacent DPs are closer than the MOR (DPs 50, 60, and 70 in the example). A new DP will be fixed at the arithmetic average of the values of DPs within each group. (This happens to be the "new" DP 60 on the

exemplary Super Tree.) The DPs which are within $\pm$MOR of the new point are now eliminated (DPs 50 and 70 in the example). To generate the next and later levels of the crude Super Tree and, from these, the levels of the Super Tree, we introduce some more concepts.

Let us define a *Path Vector* (PV) on a decision tree, starting at the root and ending at a level desired. The nodes passed through represent its components. A particular PV on the Super Tree has one or several *Equivalent Path Vectors*

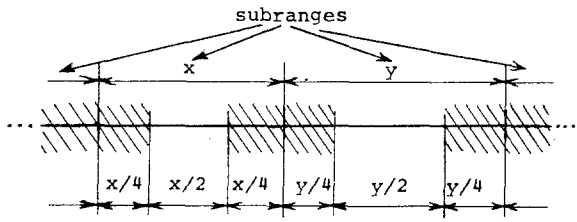

FIG. 5. The fuzzy and the certain parts of subranges.

(EPVs) on each input tree because of partially or totally overlapping subranges. (For example, the subrange 30–60 of level 1 on the Super Tree corresponds to two subranges, 0–50 and 50–100, on Input Tree A.)

The second (and further) levels of the crude Super Tree are established by forming the union of the sets of nodes at the end of all EPVs which correspond to a given PV on the Super Tree constructed so far. To illustrate the method, we have shown the computations step by step with three input trees and the final Super Tree on Fig. 3 and in Table I. The numbers in the nodes represent the values of the upper DPs of the corresponding subranges. The total range of the value of each decision variable has been "normalized" to between 0 and 100.

We define the *power of a heuristic rule* as the relative reduction in the complexity of computations, due to the use of the heuristic in question. Therefore, the power of the heuristic applied to the crude Super Tree can be defined as

$$P = \frac{C^* - C_s}{C^*}, \tag{16}$$

where C^* and C_s are the level-complexity of the crude Super Tree and of the Super Tree, respectively. C_s has to be computed the same way as the complexity of input trees using Equations (1) [or (1′)] and (2). However, we can derive a reasonable upper bound of the complexity C^* of the crude Super Tree. The worst case occurs when at any level, there are no identical subranges of decision criterion values between input trees. The number of nodes at level j of the crude

Computations for the Duper 1100 (see Text for the Explanation of Notation)

From	PV_{ST}	EPV_A	EPV_B	EPV_C	"Join" of DPs on crude ST	MIR	MOR	New DPs on ST	DPs on ST
0	(ST)	(A)	(B)	(C)	0, 30, 50, 60, 70, 100	30	15	(50+60+70)/3 =60	0, 30, 60, 100
1	(ST, 30)	(A, 50)	(B, 30)	(C, 60)	0, 40, 50, 50, 70, 100	30	15	(40+2.50)/3=47	0, 47, 70, 100
	(ST, 60)	(A, 50), (A, 100)	(B, 70)	(C, 60)	0, 40, 40, 50, 50, 100	40	20	(2.40+2.50)/4 =45	0, 45, 100
	(ST, 100)	(A, 100)	(B, 70), (B, 100)	(C, 100)	0, 30, 35, 40, 40, 60, 65, 100	30	15	(30+35+2.40)/4 =36 (60+65)/2=63	0, 36, 63, 100
2	(ST, 30, 47)	(A, 50, 50)	(B, 30, 40), (B, 30, 70)	(C, 60, 50)	0, 30, 40, 45, 60, 60, 100	15	8	(40+45)/2=43	0, 30, 43, 60, 100
	(ST, 30, 70)	(A, 50, 50) (A, 50, 100)	(B, 30, 70)	(C, 60, 50) (C, 60, 100)	0, 30, 30, 40, 45, 50, 60, 60, 100	15	8	(40+45+50)/3 =45	0, 30, 45, 60, 100
	(ST, 30, 100)	(A, 50, 100)	(B, 30, 100)	(C, 60, 100)	0, 30, 50, 50, 60, 100	30	15	(2.50+60)/3=53	0, 30, 53, 100
	(ST, 60, 45)	(A, 50, 50), (A, 100, 40), (A, 100, 100)	(B, 70, 40), (B, 70, 100)	(C, 60, 50)	0, 20, 30, 30, 40, 45, 50, 60, 60, 60, 100	15	8	(40+45+50)/3 =45	0, 20, 30, 45, 60, 100
	(ST, 60, 100)	(A, 50, 50), (A, 50, 100), (A, 100, 100)	(B, 70, 100)	(C, 60, 50), (C, 60, 100)	0, 20, 30, 30, 40, 45, 50, 60, 60, 60, 100	15	8	(40+45+50)/3 =45	0, 20, 30, 45, 60, 100
	(ST, 100, 36)	(A, 100, 40)	(B, 70, 40), (B, 100, 30), (B, 100, 60)	(C, 100, 35), (C, 100, 65)	0, 30, 40, 40, 40, 50, 50, 50, 60, 100	30	15	(30+2.40+3.50+ 60)/7=46	0, 46, 100
	(ST, 100, 63)	(A, 100, 40), (A, 100, 100)	(B, 70, 40), (B, 70, 100), (B, 100, 60), (B, 100, 100)	(C, 100, 65)	0, 20, 20, 30, 40, 40, 40, 50, 60, 60, 100	20	10		0, 20, 30, 40, 50, 60, 100
	(ST, 100, 100)	(A, 100, 100)	(B, 70, 100), (B, 100, 100)	(C, 100, 65), (C, 100, 100)	0, 20, 20, 40, 40, 50, 60, 100	20	10		0, 20, 40, 50, 60, 100

Super Tree is the algebraic sum of the nodes at this level of input trees ("join" was chosen to be the name for the operation on nodes),

$$K_j^* = \sum_{i=1}^{N} K_{ij} . \tag{17}$$

Here and later, the asterisk refers to crude Super Tree variables. Instead of just one, there are now N path vectors which belong to a given game situation. To get the jth component of each PV, the program would make

$$C_j^* = \sum_{i=1}^{N} K_{ij}^{-1} \cdot \sum_{k=1}^{K_{ij}} \log r_{ijk} = \sum_{i=1}^{N} C_{ij} \tag{18}$$

comparisons. In other words, for each level the upper bound of the level-complexity of the crude Super Tree is equal to the sum of the corresponding level-complexities of the input trees. Thus we may estimate the level-complexity C^* of the crude Super Tree by

$$C^* = \sum_{j=0}^{h} C_j^* . \tag{19}$$

We give the results of the computations performed by the program, using the input trees specified in Fig. 3. The total level-complexity of the three input trees is 3.44, 3.99, and 3.41, respectively. The sum of these, that is, the theoretical upper bound of the total level-entropy of the crude Super Tree, is 10.84. However, because the range of every decision criterion is normalized to between 0 and 100, the extreme DPs coincide and the actual total level-complexity of the crude Super Tree is only 4.87. The Super Tree itself has a total level-entropy of 2.02. The power of the heuristics used can, therefore, be calculated as (4.87–2.02)/4.87 = 0.58.

We parenthetically note that in order to calculate the power of heuristics, the program can always use the actual complexity of the crude Super Tree rather than the theoretical upper bound.

Acknowledgments

We are indebted to Mr. Paul Ruefli for many fruitful discussions and to Mr. John R. Prieur for programming the tree merging procedure. The National Science Foundation has supported one of us (N.V.F.) through Grant MCS-7624278.

Received: March 11, 1977; revised: May 5, 1978

References

Burge, W. H. (1958), Sorting, trees and measures of order, *Inform. Contr.* **1**, 181–197.

Feigenbaum, E. A. (1961), The simulation of verbal learning behavior, *in* "Proceedings of the Western Joint Computer Conference," Vol. 19, pp. 121–132.

FINDLER, N. V. (1973), Computer experiments on forming and optimizing heuristic rules, *in* "Artificial and human thinking" (A. Elithorn and D. Jones, Eds.), Elsevier, Amsterdam.

FINDLER, N. V. (1977), Studies in machine cognition using the game of Poker, *Comm. ACM* **20**, 230–245.

FINDLER, N. V., KLEIN, H., GOULD, W., KOWAL, A., AND MENIG, J. (1972), Studies on decision making using the game of Poker, *in* "Proceedings of the IFIP Congress 71," pp. 1448–1459.

FINDLER, N. V., KLEIN, H., AND LEVINE, Z. (1973), Experiments with inductive discovery processes leading to heuristics in a Poker program, *in* "Cognitive Processes and Systems" (M. Beckmann, G. Goos, and H. P. Künzi, Eds.), Springer, Berlin.

FINDLER, N. V., KLEIN, H., JOHNSON, R. C., KOWAL, A., LEVINE, Z., AND MENIG, J. (1974), Heuristic programmers and their gambling machines, *in* "Proceedings of the Association for Computing Machinery National Conference, San Diego, California," pp. 28–37.

GREEN, C. D. (1973), A path enthropy function for rooted trees, *J. Assoc. Comput. Mach.* **20**, 378–384.

JARDINE, N., AND SIBSON, R. (1971), "Mathematical Taxonomy," Wiley, New York.

MITRINOVIC, D. S. (1964), "Elementary Inequalities," Noordhoff, Groningen, Holland.

PICARD, C. (1965), "Théorie des Questionnaires," Gauthiers–Villars, Paris.

SALTON, G. (1968), "Automatic Information Organization and Retrieval," McGraw–Hill, New York.

SIMON, H. A., AND FEIGENBAUM, E. A. (1964), An information-processing theory of some effects of similarity, familiarization, and meaningfulness in verbal learning, *J. Verbal Learning Verbal Behavior* **3**, 385–396.

SIMON, H. A., AND GILMARTIN, K. (1973), A simulation of memory for chess positions, *Cognitive Psychology* **5**, 29–46.

SIMON, H. A., AND KADANE, J. B. (1976), Problems in computational complexity in artifical intelligence, *in* "Algorithms and Complexity: Recent Results and New Directions" (J. F. Traub, Ed.), Academic Press, New York.

WATERMAN, D. A. (1970), Generalization learning techniques for automating the learning of heuristics, *Artificial Intelligence* **1**, 121–170.