MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Artificial Intelligence
Memo No. 189

February 1970

CONSTRUCTION OF DECISION TREES

Edwin Roger Banks

The construction of optimal decision trees for the problem stated within can be accomplished by an exhaustive enumeration. This paper discusses two approaches. The section on heuristic methods gives mostly negative results (e.g. there is no merit factor that will always yield the optimal test, etc.), but most of these methods do give good results. The section entitled "Exhaustive Enumeration Revisited" indicates some powerful shortcuts that can be applied to an exhaustive enumeration, extending the range of this method.

CONSTRUCTION OF DECISION TREES

Edwin Roger Banks

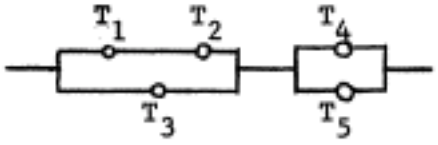
INTRODUCTION

A. The Problem

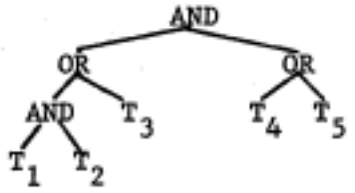
This paper considers the optimal procedure for determining whether a network of switches is open or closed. Each switch i has an a priori probability p_i of being closed and an associated cost C_i to determine the condition of the switch. The problem can also be expressed in Polish notation as, for example:

$$(\text{AND } (\text{OR } (\text{AND } T_1\ T_2)\ T_3)\ (\text{OR } T_4\ T_5)\ )$$

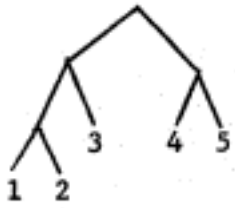
where the T_i are tests with true or false outcomes and associated p_i (of being true) and C_i. A third formulation of the problem will be used in this paper: The problem tree for the above Polish form is shown with the network representation:



NETWORK (Closed?)



a. one form.

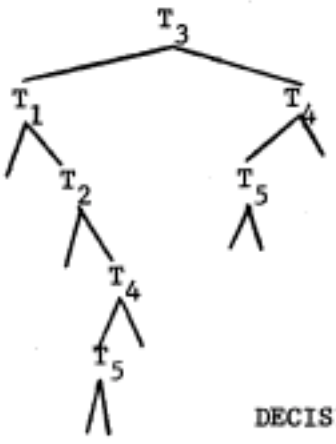


b. short form

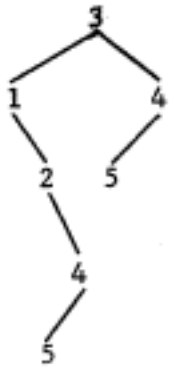
PROBLEM TREE

Bridge networks will be disallowed.

Our goal is to obtain the decision tree of minimum expected cost. A typical decision tree for the above problem tree is shown:



OR:



where / indicates open circuit; \ indicates a closed circuit.

DECISION TREE

The expected cost of this decision tree is:

$$
\begin{aligned}
& C_3 \; + \\
& p_3 \; (C_4 + q_4 \; C_5) \; + \\
& q_3 \; (C_1 + p_1 \; (C_2 + p_2 \; (C_4 + q_4 \; C_5) \;) \;)
\end{aligned}
$$

where $q = 1 - p$ is the a priori probability of being open. The above formula was obtained from the tree as follows. C_3 is the first test; it is made unconditionally. If the test results in switch 3 being closed, then the parenthesized part of the second line will become the expected cost of determining the remaining part of the circuit. And switch 3 is closed with probability p_3.

Consideration of this problem probably originated (see Berlekamp[1]) in an attempt to optimize telephone switching circuits.* Another area is problem solving where the solution to a problem can involve solving sub-problems. In our problem tree, the AND represents a division of the problem into sub-problems which jointly must be solved, and the OR, those for which the solution of any sub-problem solves the problem.

*In this problem the cost is time.

Let n be the number of switches, and X_n the number of possible decision trees. To be considered in X_n we require only that the tree has no repeated tests. (Otherwise $X_n = \infty$) Then X_n becomes:

$$X_n = n\,(X_{n-1} + 1)^2 \quad \text{and} \quad X_1 = 1$$

or

$$X_n = 1, 8, 243, 238144, 200000000000, \ldots \text{ for } n = 1, 2, 3, 4, 5, \ldots$$

This formula can be critized because it counts incomplete decision trees, but if X_n is to rule out these trees, then it becomes a function of how the n switches are interconnected. In any event, a procedure which attempts to find optimal trees by exhaustive enumeration and comparison will be practically limited to n=4 or less.

Interestingly, out of the 238000 trees for the 4-switch network, for example, there are exactly **eight** which qualify as a possibly optimal decision tree, regardless of the values of the C_i, the p_i or how connected! Results of this sort are reported in the section titled "EXHAUSTIVE ENUMERATION REVISITED".

The above results were discovered after an initial effort to apply heuristics to the problem. Several interesting theorems, including the failure theorem which shows that counter-examples exist for a broad class of heuristics, are reported in the next section.

The last section considers the technique of test-at-a-time, i. e. instead of constructing a good decision tree which is simply read in order to choose a test, how can a good test be chosen without the tree? The merit of this technique is that the decision tree is typically very large requiring storage space. (If a bushy problem tree is assumed, the average path length of the decision tree will be n/2 giving a total size of about $2^{n/2}$.)

HEURISTIC TECHNIQUES

The first class of heuristics involves the merit-factor approach. A merit factor is calculated for each switch and the largest (or smallest) merit factor determines the test. For each possible outcome of the test (open or closed) the circuit diagram is simplified and new merit factors are computed. The merit factor may be a function of the costs or probabilities of any or all of the switches and of the structure of the network itself, and other factors. We will design a few merit factors and analyze them.

A useful quantity to include in a merit-factor is the a priori probability that the entire circuit is closed. Let P designate this quantity which is easily evaluated from the p_i and the problem tree. We will also use $Q = 1 - P$ as the total probability of an open circuit. Two other useful quantities are ΔP_i and ΔQ_i. ΔP_i will represent the increase which results in P if switch i is closed. Similarly, ΔQ_i will represent the increase in Q for switch i open.

As tests are made, a plot of P against cost can be made:

The goal is the top or bottom line.

P as a Function of Tests Drawn for a Particular Decision Tree.

This diagram suggested most of the following merit factors.

1. Heuristic: make that test which gives the greatest expected change in P per unit cost. Thus we define

$$F_{1_i} = (p_i \Delta P_i + q_i \Delta Q_i) / C_i$$

2. Heuristic: make that test which gives the greatest percent change in P per unit cost:

$$F_2 = \left(\frac{p \Delta P}{P} + \frac{q \Delta Q}{Q} \right) / C$$

3. Heuristic: weight the factors of F_1 by P and Q. Thus we multiply the expected increase by P which is the probability that an increase is desired.

$$F_3 = (p P \Delta P + q Q \Delta Q) / C$$

4. Heuristic: make the test which yields the largest expected percent reduction in distance remaining to the goal:

$$F_4 = \left(\frac{p\ \Delta P}{Q} + \frac{q\ \Delta Q}{P} \right) / C$$

THEOREM I. The above heuristics are equivalent (within constants) to each other and to the simple merit factor F_5 :

$$F_5 = p\ \Delta P / C$$

To prove THEOREM I we need LEMMA I:

LEMMA I. P is individually linear in p_i. That is, for functions G and H:

$$P = G(p_1, p_2, \cdots p_{i-1}, p_{i+1}, \cdots p_n) + H(p_1, \cdots p_{i-1}, p_{i+1}, \cdots)\ p_i$$

Inductive proof of LEMMA I.

Any network (including bridge type networks) can be built by starting with one switch and two nodes and adding switches and nodes by use of two methods.

I. By connecting two existing nodes by a switch.

II. By adding a switch and a node between a switch and a node.

I. CONNECTING TWO NODES BY A SWITCH

II. INSERTING A SWITCH AND NODE BETWEEN A SWITCH AND NODE

The lemma is obviously true for a circuit with only one switch and two nodes. Consider case I. first. Let us add switch i. P' is the new probability and P is individually linear by the induction hypothesis.

$$P' = p_i\ P'' + (1 - p_i)\ P$$

P'' is a network with one less node and is therefore satisfactory. Obviously P' is individually linear. Considering case II as we add switch i:

$$P' = p_i\ P + (1 - p_i)\ P''$$

again still individually linear in p_i, Q.E.D.

Now we can prove THEOREM I. First we show the interesting equivalence

$$p_i \Delta P_i = q_i \Delta Q_i \qquad \text{for all } i.$$

Since P is individually linear in p_i we can write

$$P = G + H p_i$$

where G and H are the functions of the probabilities of the other switches. ΔP_i is the increase in P if $p_i = 1$ or

$$\Delta P_i = (G + H \cdot 1) - (G + H p_i) = H (1 - p_i) = H q_i$$

And since

$$G + H p_i = G + H (1 - q_i) = (G + H) - H q_i$$

and $$Q = 1 - P = (1 + G + H) + H q_i$$

we can show in a similar manner as above for ΔP_i that

$$\Delta Q_i = H p_i$$

Therefore

$$p_i \Delta P_i = q_i \Delta Q_i = H p_i q_i$$

Let us use X as a shorthand for $p \Delta P$ to pro ve the equivalence of our merit factors.

$$F_1 = (p \Delta P + q \Delta Q) / C = (X + X) / C = 2 X / C$$

$$F_2 = (X/P + X/Q) / C = (1/PQ) X / C \quad \text{since } P + Q = 1$$

and P and Q are initially the same for all tests.

$$F_3 = (P X + Q X) / C = X / C$$

$$F_4 = (X/Q + X/P) / C = (1/PQ) X / C$$

Also any monotonically increasing functional combination of $F_1 \ldots F_5$ will be equivalent to F_5 in the sense that it will yield the same test and hence the same decision tree.

One might expect that the time to find the largest $p\Delta P / C$ would be at least proportional to the number of switches. Thus to construct a decision tree would require a time proportional to n times the number of tests in the entire decision tree. Actually a time proportional to log n is all that is needed after the first test is chosen, if all the previous results is saved in tree form.

We have used $2^{n/2}$ as the expected size of the decision tree. (Total size of a binary tree is twice the number of end points.) The following intuitively obvious THEOREM suggests why the decision tree should be so large.

THEOREM II. There is at least one path of length n in every (complete) decision tree, i. e. it is always possible that when using the decision tree, the state of <u>every</u> switch would have to be determined to determine the state of the circuit (open or closed).

Inductive Proof of THEOREM II.

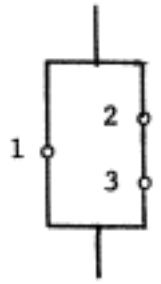
The theorem is obviously true for one switch. There are two cases--either a switch is connected in parallel or in series. If in parallel and if open, then there results a network with one fewer switch and THEOREM II applies to this. A parallel argument (pardon the pun) applies to the series connection. Thus an average depth of about n/2 results. It should be noted that for <u>any</u> technique yielding optimal trees, the time required is probably at least proportional to the size of the tree it is building giving $2^{n/2}$ as a lower bound. (It is realized that perhaps the tree contains many duplicate sub-structures in which case a time less than the above may exist.)

FAILURE THEOREM

THEOREM III. There does not exist a merit factor F_i such that the optimal decision tree is always found by picking that switch with maximum (or minimum) F. F can be a function of p_i, C_i, the problem tree structure, the other p_j ($j \neq i$), indeed of anything except the other switches' costs. F_i is restricted only to be independent of C_j for $j \neq i$.

Proof.

Consider the following network:

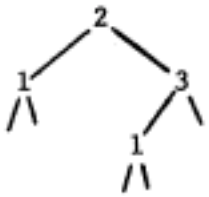


switches 1, 2, 3

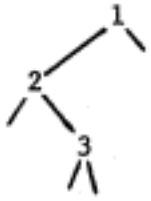
all $p_i = 1/2$

Also consider the following assignments of costs to the switches with optimal decision trees shown below.

$C_1 = 4.$
$C_2 = 1.$
$C_3 = 1.99$

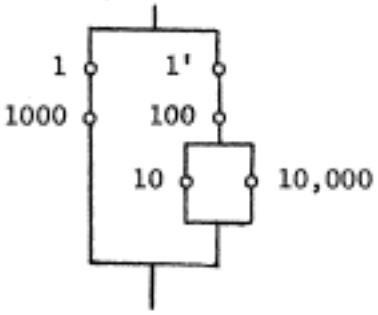


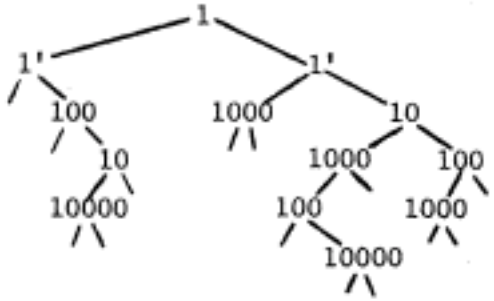
$C_1 = 4.$
$C_2 = 1.$
$C_3 = 2.01$



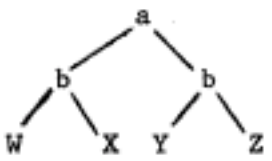
In the first case $F_2 > F_1$ while in the second case $F_1 > F_2$. But the only change was in C_3, and F_1 and F_2 were not allowed to depend on C_3.

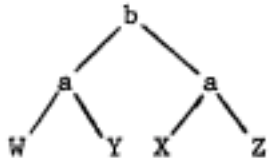
Berlekamp gave the following interesting example whose optimal decision tree is shown. Again the numbers are costs; all $p_i = 1/2$.





However, the following two trees must be equivalent where W, X, Y, and Z are substructures.





EQUIVALENT TREES

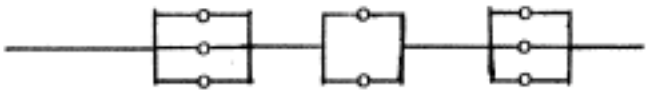
The interesting fact is that the costs 1 and 1' in the example can be varied independently over a small range while both forms of the decision tree remain equivalent.

Winston[2] has devised some heuristics for building trees that add tests and then do local improvements which can move the new tests to a position of lesser depth in the tree. His decision trees are for a more general problem in which the tests are not simple true or false, but divide a set of objects into two classes.

Reinwald and Soland[3] present an algorithm for finding the optimal tree, but Winston claims that it is only slightly better than exhaustive enumeration.

BERLEKAMP'S RESULTS

Berlekamp[1] defined a couple of merit factors but used them in a different way from our usage. He defined a parallel merit factor PMF = p_i/C_i and a series merit factor SMF = q_i/C_i and showed that for a pure parallel or pure series network, then PMF and SMF (when evaluated and ordered for all switches) would determine the optimal tree. In fact, extending this to a parallel-series (i.e. a parallel connection of pure series circuits) or series-parallel (beads on a string) network, he found the following algorithm would give the optimal tree. We illustrate here only the series-parallel case.



Example of Series-Parallel network.

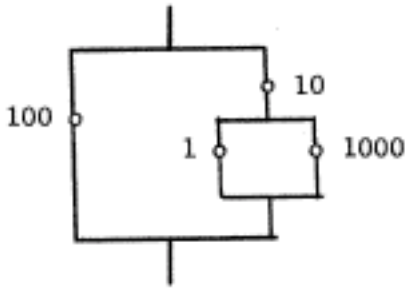
I. Replace each bead by a single switch whose cost is the expected cost of the bead and whose probability is the probability of the entire bead.

II. Calculate SMF to pick the bead for the first test.

III. Within the bead, calculate PMF to determine the switch. This switch is the first test.

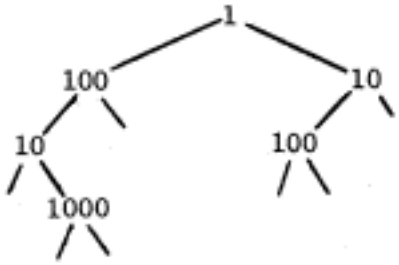
IV. Simplify the network (for both cases--open and closed switch) and start over.

Although the method always yields the optimal tree for problems whose problem tree has a depth of two, unfortunately attempts to generalize to higher depths fail. Berlekamp gave the following counter-example (depth of 3) where the numbers are the costs of the tests.



COUNTER-EXAMPLE

all $p_i = 1/2$



OPTIMAL DECISION TREE

Although the optimal tree above has an expected cost of 208.5, the application of Berlekamp's method yields a tree with expected cost of 230.25. In fact Berlekamp found a formula for how much the method costs.

BERLEKAMP'S THEOREM IV. Let T_c be the cost of the best strategy in which parallel branches a and b are looked at consecutively and T_{opt} the cost of the optimal strategy. Then

$$T_c \leq T_{opt} + (p_a/t_a - p_b/t_b)\, t_a\, (t_a + q_a\, t_b)$$

(assuming $p_a/t_a \geq p_b/t_b$). The first factor is the difference in the merit factors; the second is the cost of the first branch, and the third is the cost of the equivalent combination of the two branches.

COROLLARY. If the merit factors are equal, the branches may be combined with no loss in expected cost.

However, Berlekamp did show the following useful theorem.

BERLEKAMP'S THEOREM III. If the optimal strategy starts in a particular bead B_1, it will start at that branch of B_1 which has the highest parallel merit factor.

Attempts to generalize this theorem also failed, as shown in the next theorem.

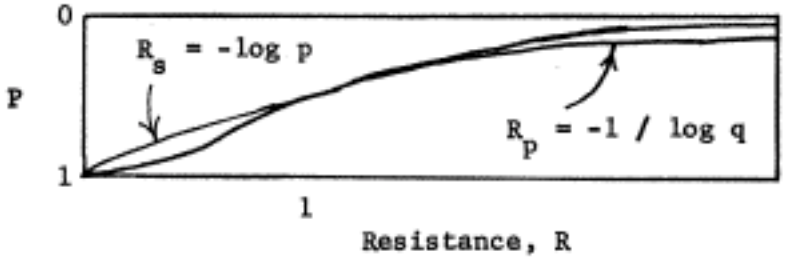
BERLEKAMP'S FALSE CONJECTURE IV. If the optimal strategy starts in a substructure N, it will start at the same place that it would have started in N alone. (Not true.)

An attempt to get the effect of the entire structure into the problem involved substituting electrical resistors for the switches. This type of operation has been known to work for some other problems. We would like to have the resistance be a function of the probability such that the best switch is found by taking that switch which has i) maximum current, ii.) maximum voltage drop or iii) maximum power dissipation per unit cost assuming a one volt applied voltage. If such a method is to work for our problem, it should satisfy the following requirements:

I. For $p = 0$, $R = \infty$ and for $p = 1$, $R = 0$

II. Resistors should combine into equivalent resistors by the parallel and series combination laws.

Unfortunately the combination laws specify the form of the equation and I. specifies the initial conditions giving two different equations depending on whether R is in series or parallel.

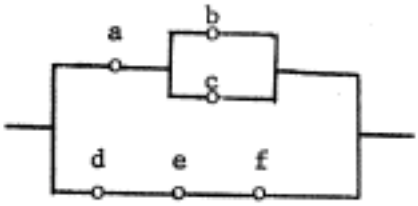


The fact that the two curves are so similar suggests that some combination of R_s and R_p should give <u>good</u> results.

EXHAUSTIVE ENUMERATION REVISITED

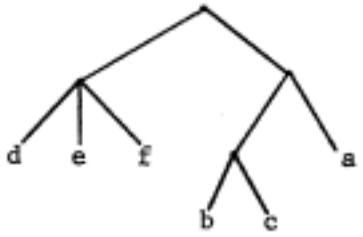
In the set of all possible decision trees for a large number of switches, n, only a very small fraction of these are possible as optimal trees. Some of these have identical costs.

The following results assume that the problem tree is slightly modified to contain the switches ordered by their (Berlekamp) merit factors at any particular level, as shown in the example.



$PMF_b \geqslant PMF_c$

$SMF_d \geqslant SMF_e \geqslant SMF_f$



(depth = 3)
PROBLEM TREE

Berlekamp's algorithm can be used on any problem whose depth is two or less. To help eliminate non-optimal trees, THEOREM IV can be used.

THEOREM IV. At any node in a decision tree, the maximum number of remaining tests assuming the outcome is a closed switch and the maximum number for an outcome of open switch are different numbers.

Proof.

In the proof of THEOREM II it was noted that one outcome gave a subproblem of exactly one less switch. But the other case must have cut off part of the circuit giving a still smaller network. For instance if the switch was connected in series and was open, then at least two fewer switches result in the subproblem.

This theorem immediately eliminates all trees that end in either of the following structures.

THEOREM V. No structure of the following form can exist as an optimal decision tree.

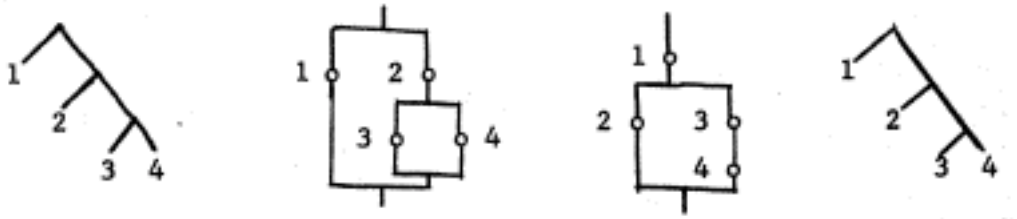
This theorem states that any two switches a and b cannot appear at the same level as a parallel connection in one case and series in another.

Argument.

In the problem tree, a and b have a youngest common ancestor which is at the parallel or series level, but whichever, it cannot change. (This theorem does not apply to bridge-type networks.)

THEOREM VI. In the 3 switch network composed of switch a in series with parallel b and c, if the SMF for a is greater than the maximum SMF for b and c, then the tests will be applied in order of SMF. Similar results hold for the dual case.

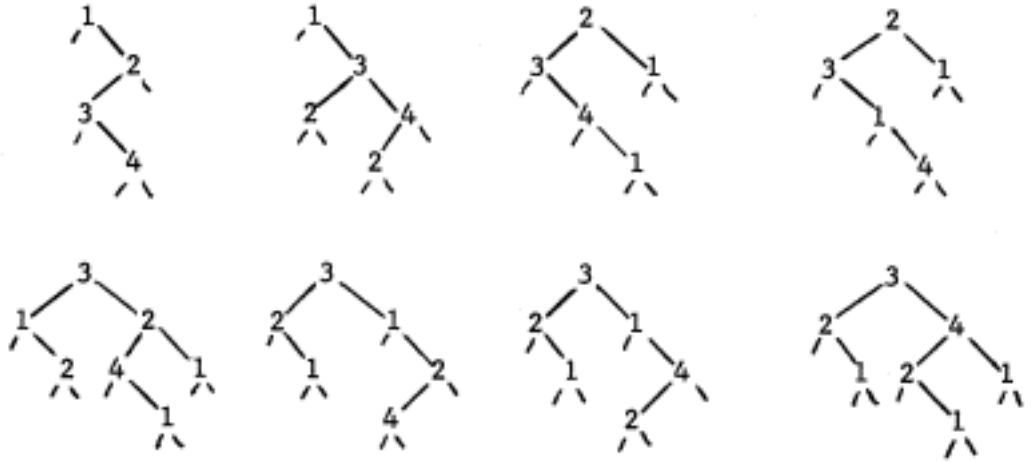
The simplest network for which Berlekamp's method fails (i. e. depth greater than two) has four switches. This is the only 4 switch network for depth of three.



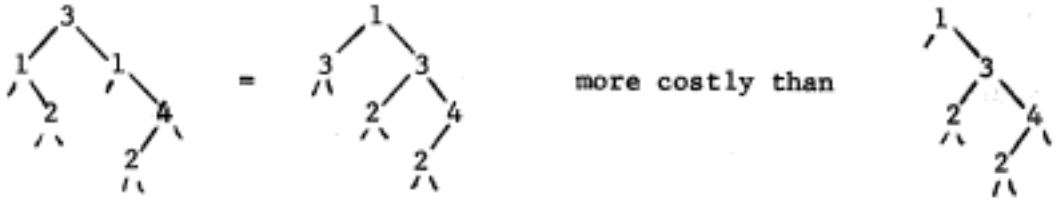
PROBLEM TREE NETWORK DUAL NETWORK PROBLEM TREE

Since the dual network is solved exactly the same as the original network giving identical decision trees, only one of each dual pair is to be considered.

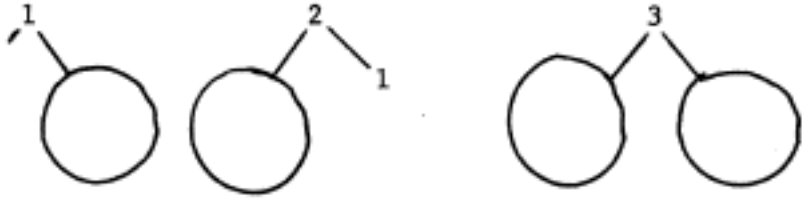
The formula for X_n (the number of decision trees) gives about 230,000 trees for the 4 switch problem. However, no more than the following eight trees need be considered. All the rest must be non-optimal.



How did these eight arise? First BERLEKAMP'S THEOREM III plus the fact that the switches are ordered by their merit factors before we start eliminates switch 4 from consideration immediately. Now consider proving, for example, that the following tree is non-optimal. We will generate a sequence of equivalent trees the last of which is more costly than one of the above eight.

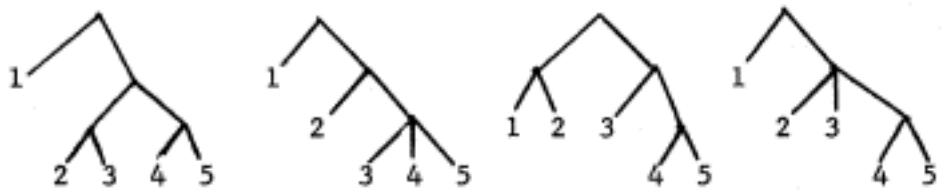


The above eight trees can be lumped into three classes where within each circle the depth is two or less and Berlekamp's method can be used to find the optimal tree.

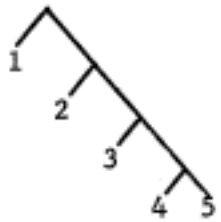


Only these three cases now need be considered. (We are now assuming that if the depth of the problem tree is less than three, Berlekamp's method is applied; but if the depth is three, then these three trees are considered.)

Going to five switches there are four meaningful problem trees of depth three and one of depth four.



FIVE SWITCH DEPTH = 3 TREES



FIVE SWITCH DEPTH = 4 TREE

Depending on which of the above problem trees is being considered, there are only 7, 5, 5, 5, or 8 trial to be made, respectively (see Appendix I). For $n = 5$, $X_n = X_5 = 2 \times 10^{11}$ approximately. Optimal trees were found for each of the listed 4 switch trees, but perhaps some of the 5 switch trees can be eliminated by further study.

Considering six switches, there are at most five ways to pick the first one since at least one way violates the theorem of Berlekamp. By THEOREM V one of the sub-problems (after the first one is made) has five switches and the other has four or less. Thus the greatest time required is proportional to $5 (8 + 3) = 55$.

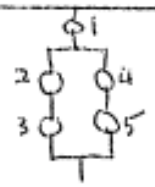
To further speed these operations, some other techniques could have been used. For example, we have seen that some decision trees give identical costs and only one member of each family should be considered. Another area for improvement concerns avoiding duplication of effort on those sub-structures which appear many times within a larger tree. Furthermore the techniques of this chapter have completely abandoned consideration of the values of the p_i and C_i of the switches. Some function of these values could probably divide the exhaustive enumeration into binary halves, greatly speeding the calculations. It is very probable that these and other pruning techniques could be developed to the point that fairly large networks (say 20 switches) could be handled in a reasonable amount of time. But if we remember that the size of a typical decision tree itself grows as $2^{n/2}$ we might find that about 20 to 30 would represent an upper bound on all techniques.

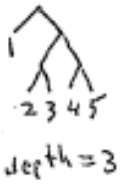
TEST AT A TIME METHOD

Since for large values of n the size of the decision tree becomes unreasonably large, it becomes desirable to have a procedure which determines the switches as the problem develops. Since the $p\Delta P/C$ method could be used (time to find the maximum value of this merit factor was proportional to log n) in a look-ahead scheme as a static evaluation function, tests could be determined very rapidly. Unfortunatly for the methods of the preceeding section, it appears that the entire tree would have to be determined to find the first test.

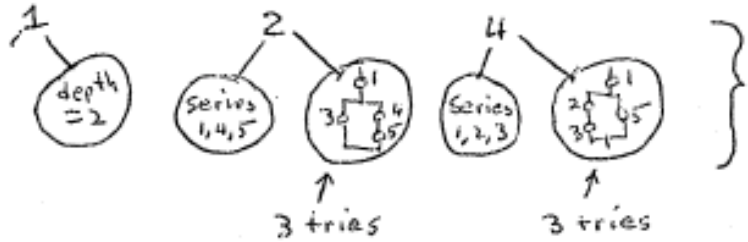
APPENDIX I

The five-switch problem: depth ≥ 3. (Duals not considered since tree is same.)



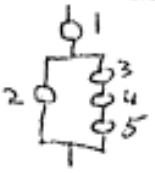


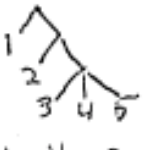
depth = 3



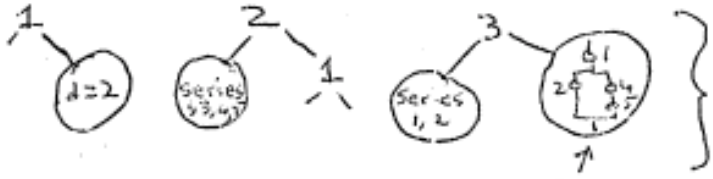
3 tries 3 tries

total = 7 trie



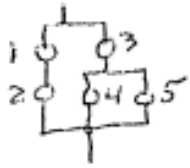


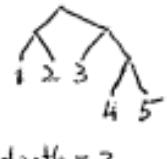
depth = 3



3 tries

total = 5 trie

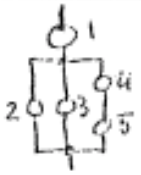


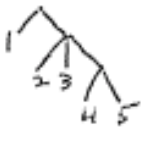


depth = 3

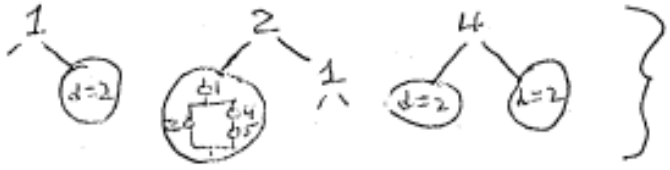
3 tries

total = 5 tries



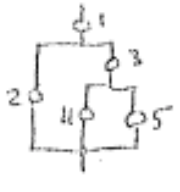


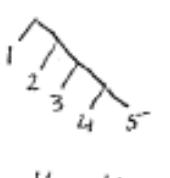
depth = 3



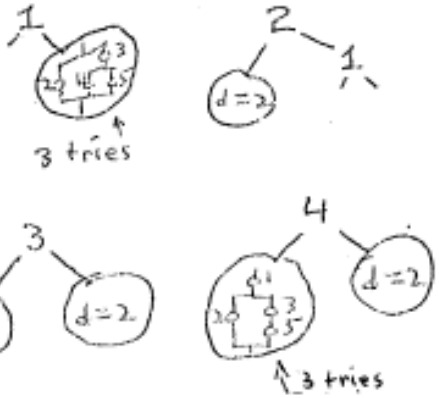
3 tries

total = 5 tries





depth = 4



total = 8 tries

BIBLIOGRAPHY

Berlekamp, Elwyn R., unpublished paper, Bell Telephone Labs., Inc., Murray Hill, N. J., dated May 31, 1966.

Winston, Patrick H., M.I.T. PROJECT MAC Artificial Intelligence Memo No. 173, March 1969.

Reinwald, Lewis and Richard Soland, "Conversion of Limited-Entry Decision Tables to Optimum Computer Programs I: Minimum Average Processing Time", JACM, July 1966.

Banks, Edwin Roger, unpublished MIT 6.544 term paper on which this paper is based.