The average complexity of depth-first search with backtracking and cutoff

by Harold S. Stone
Paolo Sipala

This paper analyzes two algorithms for depth-first search of binary trees. The first algorithm uses a search strategy that terminates the search when a successful leaf is reached. The algorithm does not use internal cutoff to restrict the search space. If N is the depth of the tree, then the average number of nodes visited by the algorithm is as low as $O(N)$ and as high as $O(2^N)$ depending only on the value of the probability parameter that characterizes the search. The second search algorithm uses backtracking with cutoff. A decision to cut off the search at a node eliminates the entire subtree below that node from further consideration. The surprising result for this algorithm is that the average number of nodes visited grows linearly in the depth of the tree, regardless of the cutoff probability. If the cutoff probability is high, then the search has a high probability of failing without examining much of the tree. If the cutoff probability is low, then the search has a high probability of succeeding on the leftmost path of the tree without performing extensive backtracking. This model sheds light on why some instances of NP-complete problems are solvable in practice with a low average complexity.

1. Introduction

This paper finds the average complexity of two kinds of depth-first search algorithms. The problem is of interest because the traditional worst-case analyses have shown that the complexity grows exponentially in the depth of the search tree. However, there exist interesting practical problems whose average complexity grows far less quickly. For example, the Traveling Salesman problem is known to be NP-complete and therefore is in the class of problems for which there exists no known algorithm whose worst-case complexity grows less than exponentially in the size of the problem. Smith [1] studied the Traveling Salesman problem and found that the average complexity of a branch-and-bound (ordered depth-first) algorithm for this problem is only $O[N^3 \ln (N)]$, where N is the number of nodes in the graph. Since the natural size of the problem is the number of edges, which grows as the square of the number of nodes, the average complexity grows less than quadratically in the size of the problem. Roth's D-algorithm [2] for generating test patterns for logic circuits discovers test patterns through an efficient backtrack search procedure. After years of extensive use, the overwhelming evidence is that each test generation cycle runs in an average time that grows linearly with the number of logic gates. The purpose of this paper is to explore a mathematical model that offers a partial explanation for that low average complexity.

In this paper we examine two different models of probabilistic behavior of a depth-first search. The objective is to estimate the average complexity of real programs. While the results from the models give an intuitive picture of the behavior of depth-first search, they are generic models and

do not model specific algorithms. Hence, particular depth-first algorithms may behave quite differently from the algorithms described in this paper, and our results will not hold for such algorithms.

The first of the two models is depth-first search without internal cutoff. This algorithm explores a decision path from the root of the search tree to a leaf node in the tree. At the leaf node, the algorithm determines whether the search is successful. If so, the search terminates. If not, the algorithm backtracks through the search tree, continues until it reaches another leaf node, and repeats this process until it terminates with success or until all of the leaves are exhausted. Our analysis shows that the average number of nodes visited is a function that varies from being linear in the depth of the tree to being exponential in the depth of the tree, depending on the probability of making good decisions on the path to a leaf.

The second algorithm exploits internal cutoff to eliminate searching of entire subtrees and terminates at the first leaf node encountered. The analysis shows that the average number of nodes visited by this algorithm tends to grow linearly in the depth of the tree rather than exponentially, regardless of the probability of cutoff. This surprising result is due to the fact that for high cutoff probability very little of the tree is actually examined, and the search has a high probability of failing. For low cutoff probability, the search tends to take successful paths, and with high probability the search succeeds in reaching a leaf node without having to examine most of the branches in the tree.

Although the number of nodes visited grows only linearly in the depth of the tree, because the cutoff computation may be a function of the depth of the tree, the average complexity of the search algorithm can grow much faster than linearly with the depth of the tree. For example, a particularly expensive way of implementing cutoff is to decide which node to visit at level i by examining all 2^i nodes at level i. In this case the cutoff computation grows exponentially in the depth of the tree, and the search is actually a breadth-first search. If the complexity of the cutoff computation is a polynomial function of the depth of the tree, then our model shows that the average complexity of the full search algorithm is a polynomial function of the depth of the tree.

Critical assumptions of the analysis are that the cutoff probability and the branching factor are uniform throughout the search tree. We do not claim that our results hold for problems where these assumptions are violated. In the N-Queens problem, for example, the branching factor near the root of the search tree is much higher than near the leaves of the search tree. Such problems may well yield average complexity that grows exponentially in the depth of the tree.

Karp and Pearl [3] have examined a related algorithm model and obtained similar results, although our analytical techniques are quite different. The methods in this paper are constructive and yield very close approximations to the exact functions that describe the search behavior as a function of tree depth, as opposed to the asymptotic values of the functions published by Karp and Pearl. Also, the methodology in this paper yields results for the average length of successful search paths, which are not directly obtainable from the theory of branching processes on which the work of Karp and Pearl is based.

The analysis is not necessarily limited to strictly depth-first searches. Branch-and-bound algorithms such as those described by Smith [1] have the property that they visit many nodes in a tree before deciding on a specific node to "expand" for further processing. Branch-and-bound algorithms can be viewed as best-first algorithms rather than depth-first algorithms. These algorithms exhibit an efficient cutoff process much like the cutoff explored in this paper, and as such the analyses contained here may be useful in characterizing these algorithms.

The results presented here are rather robust and appear to be insensitive to small perturbations in the underlying model. We show in this paper that the results do not change qualitatively when we relax the constraint that the first leaf node reached must produce a success. We also derive expressions for the variance in the number of nodes visited, and find that in several cases the variance is independent of the depth of the tree, so that the calculated value of the mean number of nodes visited is a very good estimate of the observed values in those cases. The highest growth rate for the variance is $O(N^3)$ for cases in which the average number of nodes visited grows as $O(N)$. The ratio of standard deviation to mean in these cases grows as $O(N^{0.5})$, which places a reasonably tight constraint on the observed values of the number of nodes visited, although the observed values are not as tightly constrained as for cases that have a variance that does not depend on the depth N.

The next section discusses the model with no internal cutoff. In Section 3 we describe the model that has internal cutoff. The analysis of this model appears in Section 4, and in Section 5 we describe a variation in which the leaf-success probabilities differ from the success probabilities of internal branches. Section 6 derives the variance on the functions that describe average search effort. Conclusions and recommendations appear in the final section.

2. A search strategy with termination at leaf nodes

The model that we use for this problem is illustrated in **Figure 1**. We call it the "hacker's problem" because it models the search for a password to enter a computer system illegally. To make an illegal entry, the prober guesses at each character in a password until a candidate trial is built. At this point the prober submits the candidate and is told whether the password is correct or not. In our model, we use binary characters, and we presume that the prober searches by choosing in each position the most likely bit for that

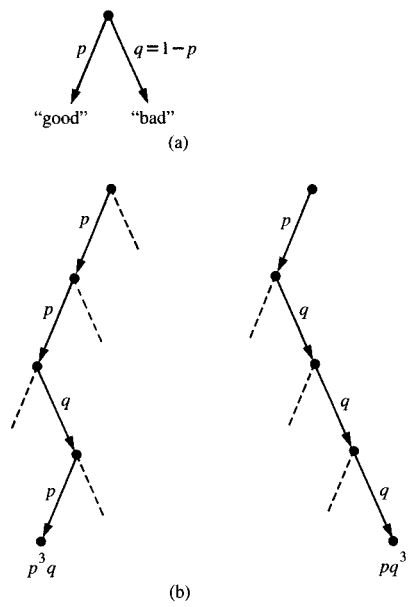

Figure 1

Probabilistic model of the tree search with no internal cutoff: (a) Probability of a good choice at a decision node; (b) Probabilities of two different paths in a search tree.

position. If a password fails, the prober backs up and changes the most recent decision. In Figure 1 each decision has a more probable path (the left-hand path), and a less probable path (the right-hand path). The search strategy starts at the root of the tree shown in the figure and progresses N levels in the tree, making N decisions along the way, one at each level. The leaves of the tree are decision points. When the algorithm reaches a leaf, it has completely characterized a possible solution (password) through the sequence of N decisions made between the root and the leaf node. The algorithm next consults an oracle to determine whether this solution is acceptable. The oracle says "yes" or "no," where the probability distribution on the answer depends only on the number of high- and low-probability branches on the path back to the root of the tree.

The detailed assumptions of this model are the following:

1. At each internal node there are two outgoing branches. The leftmost branch is traversed first. If all paths through the left descendant fail, then the search backs up and tries the right-hand branch. If all paths through the right-hand descendant fail, then the node is said to have failed, and the search returns to the ancestor of the present node.
2. At a leaf node, the algorithm consults an oracle to determine whether the search is successful. The oracle returns SUCCESS with a probability $p^{NLEFT}(1-p)^{NRIGHT}$, where $NLEFT$ is the number of left-hand branches on the path between the leaf and the root, and $NRIGHT$ is the number of right-hand branches on this path. The sum obeys the equality $N = NLEFT + NRIGHT$, where N is the depth of the tree. If the search does not succeed, the oracle returns FAIL.
3. The algorithm terminates when the search succeeds or when all leaves have been examined.

We have chosen the probability distribution to be that of N Bernoulli trials, which is not necessarily the distribution of bits in passwords but is a reasonable assumption for the general class of searches of this type in the absence of additional information for specific situations.

The search algorithm traverses the search tree by backtracking. By picking the most likely branch first at every node of the search tree, the algorithm tends to reduce the search complexity, but the algorithm is not optimal, since the cost is lower if the partial passwords are ordered by probability rank, with the more likely bit sequences appearing before the less likely bit sequences.

The model is characterized by a small set of recurrence relations. Let C_L, C_R, and C_B, respectively, be the cost of traversing a left branch, traversing a right branch, and backtracking along a branch after a failure. Let $\mathrm{av}(N)$ designate the average cost of traversing a tree of depth N. This is the cost expended on the average to search the tree of depth N, possibly to succeed or possibly to fail in the search. We also need the function $\max(N)$, which is equal to the cost of searching an entire tree of depth N, given that the search fails. The following recurrence equations describe how to compute $\mathrm{av}(N)$:

$$\mathrm{av}(1) = pC_L + q(C_L + C_B + C_R), \tag{1}$$

$$\mathrm{av}(N) = p[C_L + \mathrm{av}(N-1)] + q[C_L + \max(N-1) + C_B + C_R + \mathrm{av}(N-1)]. \tag{2}$$

The first formula states that with probability p the algorithm traverses a left branch and succeeds. Otherwise, with probability $q = 1 - p$ the algorithm traverses the left branch, fails, backtracks along the left branch, then reaches the right branch. The second equation states that the cost of a search at depth N is equal to the weighted sum of two events. The first term is the probability of traversing the left branch and succeeding in the left subtree. The cost of the second term includes the cost of taking the left branch, traversing the entire left subtree and failing, backtracking along the left branch, moving down the right branch, and searching the right subtree. The max function is used to describe the search of the left subtree because that search is not terminated early by a successful outcome, and therefore it visits the entire left subtree. The max function is given by the formula

$$\max(N) = (2^N - 1)(C_L + C_R + 2C_B). \quad (3)$$

In a tree of depth N there are $2^N - 1$ left branches and an equal number of right branches. The number of backtracks is equal to the sum of the number of left and right branches, because for each branch taken there is a corresponding backtrack.

As an example of an application of (2) and (3), consider a tree of depth 2. The average cost to search this tree can be computed by examining the cost of each of the four possible outcomes multiplied by the probability of each outcome. This is

$$\begin{aligned} \text{av}(2) = {} & p^2(2C_L) \\ & + pq(2C_L + C_B + C_R) \\ & + pq(2C_L + C_B + C_R + 2C_B + C_R + C_L) \\ & + q^2(2C_L + C_B + C_R + 2C_B + C_R + C_L + C_B + C_R). \end{aligned} \quad (4)$$

Each term in (4) describes a path to a leaf, and the probability associated with each term is the probability of terminating at that leaf. Equation (4) yields precisely the same result for av(2) as Equation (2) after substituting Equations (1) and (3), followed by algebraic reductions.

The solution to Equation (2) for $N > 1$ is given by

$$\text{av}(N) = pNC_L + q(2^N - 1)(C_L + C_R + 2C_B) - qNC_B. \quad (5)$$

Note that both Equations (4) and (5) reduce to

$$\text{av}(2) = 2pC_L + 3q(C_L + C_R) + 4qC_B. \quad (6)$$

For values of p near unity, Equation (5) grows linearly in the number of levels and tends to succeed on the leftmost branches of the tree. For small values of p the second term becomes large, and the complexity tends to grow exponentially in the number of levels. It is this aspect of tree search that makes tree searches appear to be intractable for moderate numbers of levels. Plots of Equation (5) for various values of N and p appear in **Figure 2**. All costs are assumed to be unit costs in the plot. The straight lines on the semilog plot show an exponential growth as the depth increases, but for values of p very close to 1 the search complexity for shallow trees grows linearly with depth, as indicated by the logarithmically shaped portion of the curves. A very good guesser can achieve linear average search time for shallow trees, but as the search tree deepens, the search time becomes exponential.

This analysis suggests that some depth-first searches could be very efficient on the average in spite of the exponential bound on worst-case complexity. In reality, the strategy considered here is not generally acceptable because of the potential for exponential complexity. The next section shows how a strategy with internal cutoff is vastly superior on the average to the strategy used here.

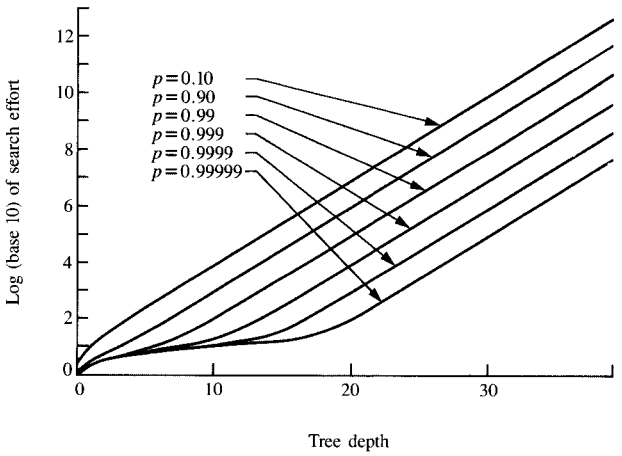

Figure 2

Expected number of nodes visited for backtrack search without internal cutoff.

3. Depth-first search with internal cutoff

In this section we examine the effects of internal cutoff and present a model in which the average number of nodes visited grows only linearly with the depth of the tree. (Complexity may grow at a rate faster than linear if the work per node explored is a function of the depth of the tree.)

The tree search considered in the previous section benefits from cutoff at success, but the strategy can do extensive computation if the successful outcome terminates at a leaf far from the left side of the tree. This search suffers from the problem that an early selection may be entirely wrong, but the algorithm continues to explore the entire subtree beneath that selection. The algorithm outline is a nondeterministic program with the following structure:

```
Choose attribute 1;
Choose attribute 2;
...
Choose attribute N;
If Oracle(attributes) then succeed else fail;
```

The Oracle procedure examines the set of attributes selected and returns a value of TRUE or FALSE depending on whether the set describes a pattern sought. To reduce the possibility of exponential search, it is general practice to seek a strategy with internal cutoff. This strategy leads to a nondeterministic program with a slightly different structure, as indicated below:

```
Choose attribute 1;
If not Oracle(attributes) then fail;
```

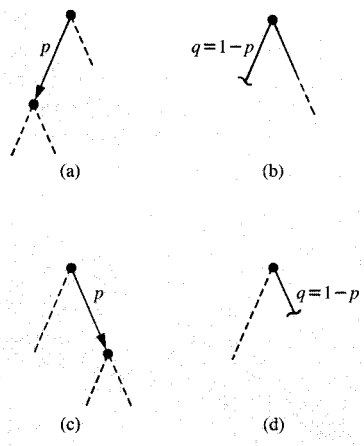

Figure 3

Probabilistic model of search with internal cutoff: (a) Success on left branch; (b) Cutoff on left branch; (c) Success on right branch; (d) Cutoff on right branch.

Choose attribute 2;
If not Oracle(attributes) then fail;
. . .
Choose attribute N;
If Oracle(attributes) then succeed else fail;

The Oracle procedure in this form of the program accepts a partial list of attributes. A FALSE result indicates certainty that no pattern exists for the corresponding selection of attributes. However, a TRUE result is returned when the Oracle is unsure of the outcome or when the Oracle knows with certainty that patterns with the selected attributes exist. The search must continue after a TRUE result is observed until all attributes are selected. Only after all are selected does a TRUE response from the Oracle indicate certainty.

Smith [1] examines a branch-and-bound algorithm similar to the latter algorithm. His oracle procedure helps to guide the search down the most likely path, and when the procedure reaches a leaf for the first time, the search terminates successfully. The total complexity of such a search may grow at a rate faster than linear in the depth of the tree because the consultations of the oracle may require computations that are in turn functions of the size of the problem. For Smith's approach the number of nodes of the search tree visited grows linearly in the depth of the tree search, at the rate $O[N \ln(N)]$, where N is the number of cities to visit in the Traveling Salesman problem. The search grows only linearly in the depth of the search tree because the search is conducted over $N!$ permutations of N cities, and a search tree with $N!$ leaves has a depth that grows at least as fast as $O[\log(N!)] = O[N \ln(N)]$.

Roth's D-algorithm [2] is similar to Smith's algorithm in that it conducts a backtracking search with cutoff until it reaches a leaf node. If it reaches a leaf node in the search tree, the search terminates successfully. Unlike Smith's algorithm, Roth's algorithm does not have to visit every element in the database to reach a successful termination, whereas Smith has to examine every city in the city graph to meet the specifications of the problem. Roth's D-algorithm may find an input combination that exercises a particular internal node properly without necessarily visiting each logic element in a circuit.

Both Roth's algorithm and Smith's algorithm have exponential upper bounds and the corresponding problems are NP-complete. They both terminate when they reach a leaf node. The fact that only one leaf needs to be examined is crucial to the analysis that follows. In this case the procedure that determines whether or not to cut off a search is the deciding factor in the nonexponential average number of nodes visited by the algorithm. If cutoff can be decided efficiently, then on the average the entire algorithm runs efficiently.

Karp and Pearl [3] have studied a slightly different problem and report results that are close to those reported here. Although they use a different search strategy, their results are identical to ours for the case in which the search fails, since the nodes actually explored by their algorithm are the same ones explored in our model. The approaches followed are quite different, however, and the methods described here yield nearly exact formulas as opposed to bounds and asymptotic formulas reported by Karp and Pearl.

To begin the analysis, let us model the effects of internal cutoff as illustrated in **Figure 3**, which shows one node of a search tree. In this case, the search makes a selection of an attribute and consults an oracle. With probability p the oracle returns TRUE and the search proceeds into the subtree, as indicated in Figure 3(a). With probability $q = 1 - p$ the oracle returns FALSE and the search is terminated on the left branch of the node without any further computation. This is shown in Figure 3(b). When the search terminates unsuccessfully on the left side of the node for any reason, the search continues on the right side of the tree. We call p the *survival* probability and q the *cutoff* probability. Figures 3(c) and 3(d) show the corresponding cases occurring on the right side. Although this model is similar to the model in the preceding section, the meaning of the probability parameter p has changed. In the prior model, p modeled the likelihood of the left branch being successful as compared to the right branch. In this model, there is no difference in probabilities between right-hand and left-hand branches. The probability parameter expresses the likelihood of early cutoff, not the asymmetry of the search.

The probability model requires several functions to describe the average cost of a search of a tree of depth N. Let

$F(N)$ and $S(N)$, respectively, denote the probability of failure and success in searching a tree of depth N. These functions are described recursively by the following equations:

$$F(1) = q^2, \tag{7}$$

$$F(N) = [q + pF(N-1)]^2, \tag{8}$$

$$S(1) = p + pq = 1 - F(1), \tag{9}$$

$$S(N) = pS(N-1) + p[q + pF(N-1)]S(N-1) = 1 - F(N). \tag{10}$$

Equation (7) says that a failure at depth 1 occurs if there is a failure on both branches. Similarly, Equation (8) says that a failure at depth N occurs if a branch fails (with probability q) or if the branch succeeds and the tree in the next lower level fails [with probability $pF(N-1)$]. Moreover, a failure must occur on both the left and right branches so that the probabilities of failure are multiplied, whence comes the square in the formula. Equations for $S(N)$ can be derived by similar arguments. Later in this section we explore analytical results concerning the growth of these functions.

We are interested in obtaining the average work performed per search. If we let $path_i$ denote one possible search path, p_i be the probability of that path, and $length(path_i)$ be the number of nodes on $path_i$, then the average number of nodes visited during a search is given by

$$L(N) = \sum_i p_i length(path_i), \tag{11}$$

where the summation is taken over all tree paths. We have adopted a unit cost for the traversal of a branch in this model, rather than carrying the costs of left and right branches separately, and the cost of backtracking is ignored because the extra complication makes the discussion less clear while not adding materially to the nature of the results. Because it is convenient to deal with failure paths and successful paths separately, we break Equation (11) into two separate summations, and this requires the definition of two new functions. Let $L_F(N)$ and $L_S(N)$, respectively, designate the contributions to Equation (11) from searches that end in failure and from searches that end in success. The formula for calculating L_F is given by

$$L_F(N) = \sum_{i \in FP(N)} p_i length(path_i), \tag{12}$$

where the set $FP(N)$ is the set of all paths in a tree of depth N along which a search of the tree fails. The function $L_S(N)$ is given by

$$L_S(N) = \sum_{i \in SP(N)} p_i length(path_i), \tag{13}$$

where the sum is taken over the successful paths.

To compute the values of $L_F(N)$ and $L_S(N)$, we may use the recursive formulas

$$L_F(1) = 2q^2, \tag{14}$$

$$L_F(N) = 2q^2 + 2pq \sum_{i \in FP(N-1)} p_i[2 + length(path_i)] + p^2 \sum_{i \in FP(N-1)} p_i \sum_{j \in FP(N-1)} p_j[2 + length(path_i) + length(path_j)], \tag{15}$$

$$L_S(1) = p + 2pq, \tag{16}$$

$$L_S(N) = p \sum_{i \in SP(N-1)} p_i[1 + length(path_i)] + p^2 \sum_{i \in FP(N-1)} p_i \sum_{j \in SP(N-1)} p_j[2 + length(path_i) + length(path_j)] + pq \sum_{j \in SP(N-1)} p_i[2 + length(path_i)]. \tag{17}$$

Equation (14) states that the unique failure path for a tree of depth 1 has length 2 and probability q^2. The recursive equation (15) states that there are three different kinds of internal failure paths. The shortest one has immediate cutoff on both branches so it has length 2 and probability q^2. The next type of path has immediate cutoff on one branch, but succeeds on the other immediate successor, only to fail in the tree of depth $N-1$ below. The last term describes failure paths that succeed on both immediate successor branches and fail in their respective subtrees of depth $N-1$. Note that the probability of the last term is p^2 and the probability of the middle term is $2pq$. The factor of 2 in the middle term arises because there are two different ways to produce such paths—one with an immediate failure on a left successor and one with an immediate failure on a right successor.

Equation (16) follows from the fact that a search of length 1 can succeed on the left branch with probability p, and a search of length 2 can succeed on the right branch with probability qp. The most complicated equation is (17). This equation sums the cost of a successful search of the left subtree with the cost of a successful search of the right subtree. Note that the last two terms account for the two ways a search can succeed in the right subtree. The second term accounts for the case in which the left branch survives and the tree beneath it fails. The last term accounts for the case in which the left branch fails and the search of the left tree is cut off immediately.

Equations (15) and (17) are very difficult to evaluate as written, and exact analytic solutions of the equations in this form are out of the question. Fortunately, both (15) and (17) simplify. By regrouping terms within the summations and using the identities

$$F(N) = \sum_{i \in FP(N)} p_i, \tag{18}$$

$$S(N) = \sum_{i \in SP(N)} p_i, \tag{19}$$

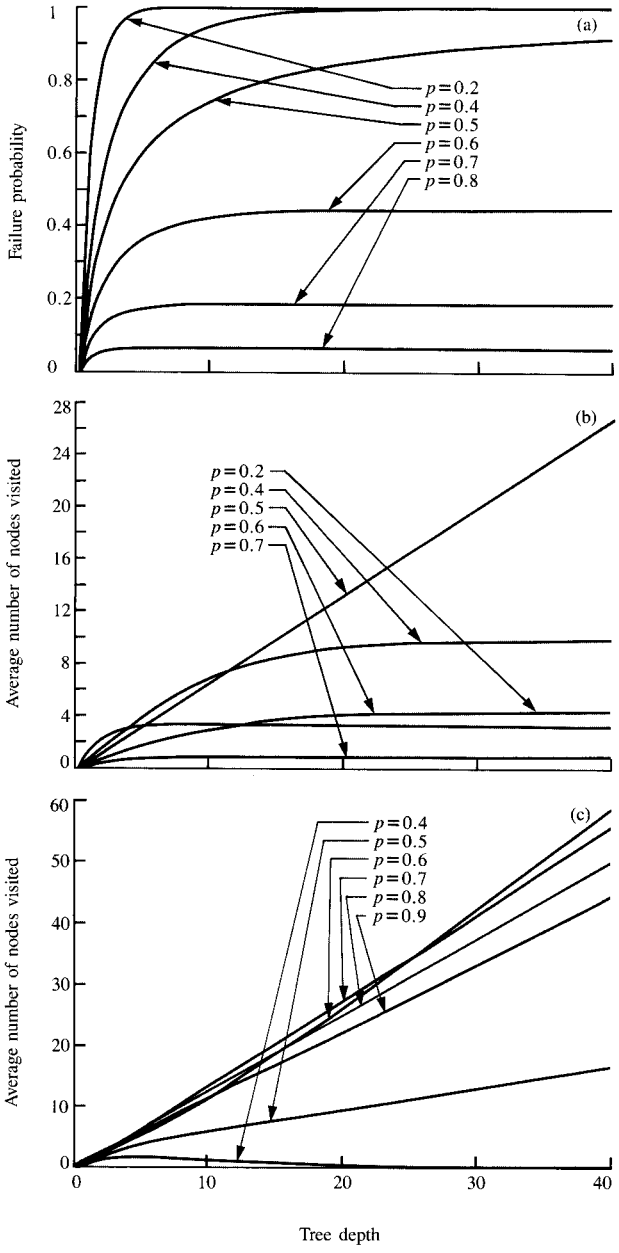

Figure 4

Statistical functions for average search complexity. (a) Probability of search failure as a function of internal branch-survival probability p. (b) Unconditional expected number of nodes visited on paths that lead to search failure. (c) Unconditional expected number of nodes visited on successful search paths.

we obtain the two recursion equations below that are readily evaluated:

$$L_F(N) = 2q^2 + 4pqF(N-1) + 2pqL_F(N-1) + 2p^2[F(N-1)^2 + F(N-1)L_F(N-1)], \quad (20)$$

$$L_S(N) = p[S(N-1) + L_S(N-1)] + p^2[2F(N-1)S(N-1) + S(N-1)L_F(N-1) + L_S(N-1)F(N-1)] + pq[2S(N-1) + L_S(N-1)]. \quad (21)$$

Data for the average search complexity for failures and successes are plotted in **Figure 4**. Figure 4(a) gives the probability for a search to fail as a function of depth and survival probability. Note the difference in behavior of the curves above and below the critical value of 0.5 for survival probability. Figures 4(b) and 4(c) plot the average number of nodes visited, respectively, for searches that fail and for searches that succeed. Note that for all values of the cutoff probability the average length of a search grows at most linearly in N. The sum of the plots in Figures 4(b) and 4(c) is the total average search complexity. Recall from Equation (11) that the probabilities used in Equations (18) and (19) are not conditioned on the success or failure of a search path. Hence, the expectations in Equations (20) and (21) are unconditional expectations, and for this case the average number of nodes visited for successful searches can be less than the depth of the tree. However, when we use conditional probabilities in Equation (18), the conditional expectation corresponding to Equation (21) is at least as large as the tree depth. This is plotted in **Figure 5** together with the conditional expectation for the failure effort.

The next section derives approximate solutions to Equations (20) and (21) to show their asymptotic behavior.

4. Analytic approximations

The goal of this section is to derive analytic solutions to the recurrences (20) and (21). Some of the results derived below can be obtained from the theory of branching processes [4]. In particular, the failure probability (7) and (8) and failure-path length (20) are explained by the theory. However, success-path length (21) is not a direct consequence of the theory because successful paths are not symmetric with respect to the root of a tree, and symmetry is required for the published results of the theory. We develop a new derivation technique that solves both (20) and (21). Because these equations depend upon search failure probability, we begin by deriving asymptotic solutions to Equation (8) and find the rate of convergence to those solutions.

The failure probability function $F(N)$ satisfies the nonlinear recurrence equation (8). Since $0 \leq F(N) \leq 1$, we know that $F(N)$ must approach a limit point or limit cycle as N becomes large. The possible limit points are values of F that solve the equation

$$F = (q + pF)^2. \quad (22)$$

There are two solutions:

$$F = 1 \tag{23}$$

and

$$F = q^2/p^2. \tag{24}$$

It is not difficult to show that successive iterations of Equation (22) converge to the smaller of the roots given in Equations (23) and (24) (cf. Harris [4]).

Figure 6 shows a plot of the limit points of $F(N)$ as a function of p, the survival probability. For values of survival probability less than or equal to 0.5, the limiting probability of search failure is unity, since the expected number of surviving branches beneath a node is less than 1 and decreases exponentially at each successive level in the tree. For greater values of survival probability, the probability of search failure falls away from 1, as shown in Figure 6. In this probability region the expected number of live branches beneath a node increases exponentially in the depth of the tree. This derivation agrees with that of Mullikin [5], who proves the following result. Let F_∞ be defined by

$$F_\infty = \lim_{N \to \infty} F(N)$$

and let $f(s)$ be the generating function whose coefficient of s^i is the probability of having i nodes active at the first generation of the branching process. In this case,

$$f(s) = q^2 + 2pqs + p^2s^2. \tag{25}$$

Mullikin [5] shows that F_∞ is the smallest nonnegative root of $f(s) = s$ and that

$$\begin{aligned} F_\infty &< 1 \quad \text{if} \quad f'(1) > 1, \\ &= 1 \quad \text{if} \quad f'(1) = 1 \text{ and } f \text{ is nonlinear}, \\ &= 0 \quad \text{if} \quad f'(1) = 1 \text{ and } f \text{ is linear}, \\ &= 1 \quad \text{if} \quad f'(1) < 1. \end{aligned} \tag{26}$$

Note that $f'(1) = 2p$ and f is nonlinear in its argument s if $p > 0$.

It is not surprising that the searching effort grows less than exponentially when the survival probability is less than 0.5. Regardless of the true depth of the search tree, the region of the tree visited during a search approaches a finite limit. What is less obvious is that the search space does not blow up for high values of p.

From an intuitive point of view, as the survival probability goes to 1, the search is more likely to succeed in reaching a leaf with a minimum of backtracking. And since this type of search terminates with certainty when it reaches a leaf, the search terminates before it expends a great deal of effort in visiting much of the tree. In our experiments with an implementation of Roth's D-algorithm, we found exactly this behavior. Most of the searches terminated quickly because the survival probability was essentially near 1. When we deliberately inserted untestable elements into a logic

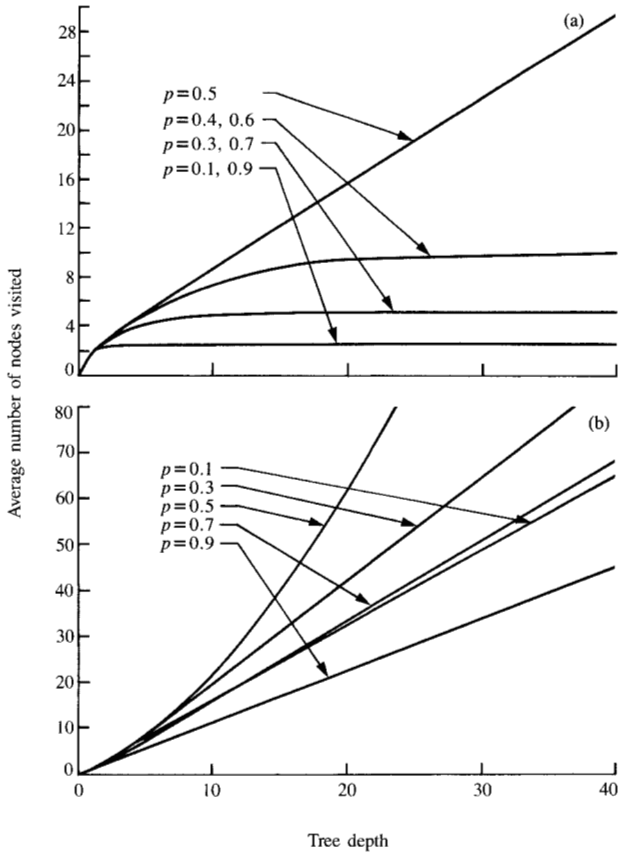

Figure 5

Conditional statistics functions. (a) Average number of nodes visited on paths that fail, given that the search fails. (b) Average number of nodes visited on paths that succeed, given that the search succeeds.

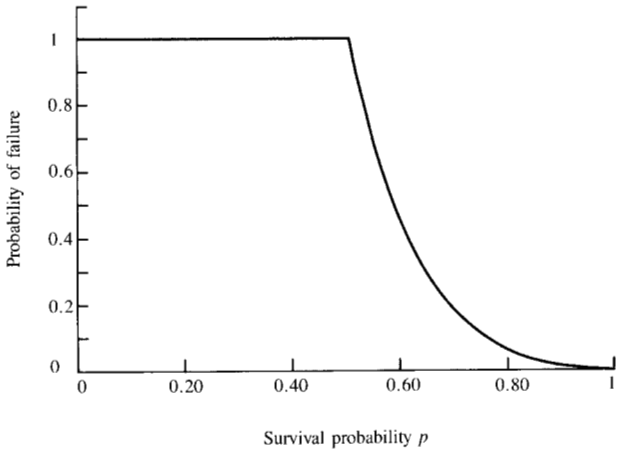

Figure 6

Search-failure probability for trees of infinite depth as a function of the branch-survival probability p.

circuit so as to prevent a successful termination, the algorithm continued to execute rather quickly, except that the algorithm generally performed more work than in the cases that yielded successful termination. We observed that the algorithm behaved as if the probability of survival were rather low in these cases, and therefore early cutoff eliminated lengthy searches.

In order to obtain approximations to functions that describe the average search complexity, we shall need to know the rate at which $F(N)$ converges to its asymptotic limit. The plot in Figure 4(a) suggests that the convergence is very fast except for $p = 0.5$. Indeed, this case is exceptional and exhibits the slowest convergence. Let

$$F(N) = r - \varepsilon_N, \tag{27}$$

where r is the asymptotic limit of $F(N)$, i.e., the smaller of the roots given in Equations (23) and (24). It follows that

$$r - \varepsilon_{N+1} = r - 2p(q + pr)\varepsilon_N + p^2\varepsilon_N^2. \tag{28}$$

From (28), when $F(N)$ increases monotonically to its asymptote, the following difference equation holds:

$$\varepsilon_{N+1} = 2p(q + pr)\varepsilon_N - p^2\varepsilon_N^2. \tag{29}$$

For the special case of $p = 0.5$, Equation (29) can be rewritten

$$\varepsilon_{N+1} = \varepsilon_N - \frac{\varepsilon_N^2}{4}. \tag{30}$$

Equation (30) is related to the equations studied by Aho and Sloane [6], who found that solutions grow at a doubly exponential rate. Their studies, however, did not treat the equation for the range of constants of interest to us, where the solution converges asymptotically to a finite limit. Franklin and Golomb [7] studied a closely related recurrence equation and discovered an application for the enumeration of trees. Their studies also explored solutions in regions that are not of interest to us in this paper. Gottlieb and Schwartz [8] show that Equation (30) describes the bandwidth of a multistage interconnection network. Although they evaluate the recursion in their paper, they do not present a solution to the recurrence. Kruskal and Snir [9] give a detailed solution of the recurrence $\varepsilon_{N+1} = 1 - (1 - \varepsilon_N/k)^k$, and the solution for $k = 2$ corresponds to our problem for $p = 0.5$. The approach we use here solves the problem for all p and gives average path lengths not directly computable with the approach used by Kruskal and Snir. It is similar in spirit to the solution of almost-linear recurrences described by Purdom and Brown [10] and Greene and Knuth [11] in that it finds an approximate solution that is perturbed by a small amount to yield the true solution. In the present case, our solution is asymptotically exact, and extremely accurate but not exact for small values of N.

Our approach is to find a continuous equation that is easily solvable whose solution approximates the solution to the discrete equation. We proceed by putting (30) into difference form, and then move to differential form. From (30), we find

$$\varepsilon_{N+1} - \varepsilon_N = -\frac{\varepsilon_N^2}{4}. \tag{31}$$

Let $\hat{\varepsilon}_N$ be a continuous variable that approximates ε_N. Using $\hat{\varepsilon}_N$ in the continuous analog of (31) yields the equation

$$\frac{d\hat{\varepsilon}_N}{dN} = -\frac{\hat{\varepsilon}_N^2}{4}, \tag{32}$$

whose solution is

$$\hat{\varepsilon}_N = \frac{4}{(N + C)}, \tag{33}$$

where C is a constant of integration. This function converges very slowly to zero.

To compare this result with the result obtained from Harris [4], we have

$$\lim_{N\to\infty} N[1 - F(N)] = \frac{2}{\sigma^2},$$

where σ^2 is the variance in the number of immediate successors of a node. Since $p = 0.5$, the variance for each successor is the variance for a single Bernoulli trial, which is $pq = 0.25$. Because there are two successors, $\sigma^2 = 2pq = 0.5$. Therefore Harris' limit for $1 - F(N)$ is $4/N$, which agrees with (33).

Kruskal and Snir [9] obtain the solution

$$F(N) = \frac{4}{N}\left[1 - \frac{\ln(N)}{N} + O\left(\frac{1}{N}\right)\right].$$

For $p \neq 0.5$, the continuous form of Equation (29) is

$$\frac{d\hat{\varepsilon}_N}{dN} = -\{[1 - 2p(q + pr)]\hat{\varepsilon}_N + p^2\hat{\varepsilon}_N^2\}.$$

For $p < 0.5$ replace $q + pr$ with unity, and find

$$\frac{d\hat{\varepsilon}_N}{dN} = -[(1 - 2p)\hat{\varepsilon}_N + p^2\hat{\varepsilon}_N^2].$$

This equation can be solved by separating variables to place it in the form

$$d\hat{\varepsilon}_N\left[\frac{1}{\hat{\varepsilon}_N} - \frac{p^2}{(1 - 2p) + p^2\hat{\varepsilon}_N}\right] = -(1 - 2p)dN. \tag{34}$$

Its solution is

$$\ln(\hat{\varepsilon}_N) - \ln[(1 - 2p) + p^2\hat{\varepsilon}_N] = -(1 - 2p)N + C(p), \tag{35}$$

where $C(p)$ is a function that depends on p but not on N. The continuous solution is only an approximation of the discrete solution, so the discrete solution has additional terms that do not appear in (35). Using the techniques of Purdom and Brown [10] for solutions to almost-linear equations, we can improve the approximation to the discrete

solution as given below:

$$\ln(\varepsilon_N) - \ln[(1 - 2p) + p^2\varepsilon_N] = N\ln(2p) + C(p). \quad (36)$$

Note that

$$\ln(2p) = \ln[1 - (1 - 2p)] = -(1 - 2p) + O[(1 - 2p)^2],$$

so that the difference between Equations (35) and (36) can be attributed to higher-order terms in the discrete solution that are absent in the continuous solution. Equation (36) is an extremely accurate approximation, and differs from the exact solution by terms that quickly go to zero as N increases. To solve Equation (36) for ε_N we first exponentiate to obtain

$$\frac{\varepsilon_N}{(1 - 2p) + p^2\varepsilon_N} = K(p)(2p)^N,$$

where $K(p)$ is a function independent of N. Solving for ε_N yields

$$\varepsilon_N = \frac{K(p)(2p)^N(1 - 2p)}{1 - p^2K(p)(2p)^N} \quad \text{for} \quad p < 0.5, \quad (37)$$

which indicates that the convergence is exponential in N. For $p > 0.5$ the equation has the similar form

$$\varepsilon_N = \frac{K(p)(2q)^N(1 - 2q)}{1 - p^2K(p)(2q)^N} \quad \text{for} \quad p > 0.5, \quad (38)$$

which again indicates that convergence is exponential in N. Therefore the only case for which convergence is not exponential is the case $p = 0.5$. This is shown graphically in Figure 4(a).

An approximation to the function $K(p)$ is easily obtained from Equations (37) and (38) by setting $N = 1$ and equating the expressions in those equations to the expressions for ε_1. After making the substitutions and solving for $K(p)$ we obtain

$$K(p) = \frac{(1 - q^2)}{2p(1 - 2p) + 2p^3(1 - q^2)} \quad \text{for} \quad p < 0.5 \quad (39)$$

and

$$K(p) = \frac{q^2(1 + p)}{2p^2(1 - 2q + q^3 + pq^3)} \quad \text{for} \quad p > 0.5. \quad (40)$$

When these expressions for $K(p)$ are substituted back into (37) and (38), the resulting equations yield extremely accurate approximations for ε_N. This convergence is predicted by Harris [4, Sec. 8.3] in his formula

$$\lim_{N\to\infty} \varepsilon_N[f'(F_\infty)]^{-N} = \frac{F_\infty - s}{1 + (F_\infty - s)\alpha(s)} \quad |s| < 1, \quad (41)$$

where $f(s)$ is the generating function defined by Equation (25) and $\alpha(s)$ is a function of s that is analytic in the region $|s| < F_\infty$ and bounded for $|s| \le 1$. Set $s = 0$ in Equation (41) and note that

$$f'(F_\infty) = 2p(1 - p) + 2p^2F_\infty, \quad (42)$$

Then, by using the values from (23) and (24) for F_∞, we discover

$$\varepsilon_N \underset{N\to\infty}{\simeq} \frac{(2p)^N}{1 + \alpha(0)} \quad \text{for} \quad p < 0.5 \quad (43)$$

and

$$\varepsilon_N \underset{N\to\infty}{\simeq} \frac{q^2(2q)^N}{p^2 + q^2\alpha(0)} \quad \text{for} \quad p > 0.5 \quad (44)$$

These equations are similar to (37) and (38) in form, but because $\alpha(0)$ is not given explicitly, Harris' equations cannot be directly compared to (37) and (38). Also, note that Equations (37)–(40) are valid approximations for all N, rather than valid only in the limit.

Now we can return to the original goal, the derivation of the asymptotic estimates of average complexity. First, let us reexamine Equation (20). This can be rewritten as

$$L_F(N) = 2F(N) + 2p[q + pF(N - 1)]L_F(N - 1). \quad (45)$$

Instead of using the exact value of $F(N)$ we can replace $F(N)$ with its asymptotic value and obtain an approximation for Equation (45) that yields good estimates for $L_F(N)$. Exact expressions for the solution of Equation (45) can be obtained by using techniques described by Purdom and Brown [10] for the solution of almost-linear recurrences. The leading terms of the solution are obtained by solving the linear recurrence, and the higher-order terms vanish as N increases. The accuracy of the approximation depends on how quickly $F(N)$ converges to its asymptotic limit. Numerical studies have shown that the approximations are excellent even for values of N as small as 10. Assuming that p is less than 0.5, we substitute Equation (23) for $F(N)$, and Equation (45) becomes

$$L_F(N) = 2 + 2pL_F(N - 1),$$

which has the solution

$$L_F(N) = (2p)^{N-1}L_F(1) + 2\left[\frac{1 - (2p)^{N-1}}{1 - 2p}\right]$$

$$\simeq \frac{2}{1 - 2p}. \quad (46)$$

For $p > 0.5$ we substitute Equation (24) in (45) and obtain

$$L_F(N) = 2\frac{q^2}{p^2} + 2qL_F(N - 1). \quad (47)$$

The solution to this equation is

$$L_F(N) = (2q)^{N-1}L_F(1) + 2\left[\frac{1 - (2q)^{N-1}}{1 - 2q}\right]\left(\frac{q^2}{p^2}\right) \quad (48)$$

$$\simeq \left(\frac{2}{1 - 2q}\right)\left(\frac{q^2}{p^2}\right).$$

The curves in Figures 4(b) and 4(c) exhibit the predicted behavior for $p \neq 0.5$ because $F(N)$ converges sufficiently fast for its approximations to be rather good. For $p = 0.5$ we must look specifically at the rate of convergence of $F(N)$. Substituting Equations (27) and (33) into (45) and setting p equal to 0.5 yields the equation

$$L_F(N) = 2 - \frac{8}{N + C} + \left(1 - \frac{2}{N - 1 + C}\right) L_F(N - 1). \quad (49)$$

By putting this into difference form similar to Equation (31), we obtain

$$L_F(N) - L_F(N - 1) = 2 - \frac{8}{N + C} - \frac{2L_F(N - 1)}{N - 1 + C}. \quad (50)$$

Proceeding as before, we move to the continuous solution by solving

$$\frac{dL_F(N)}{dN} = 2 - \frac{8}{N + C} - \frac{2L_F(N - 1)}{N - 1 + C}. \quad (51)$$

The solution to this equation approaches a linear function as N becomes large. To find that linear function, substitute $L_F(N) = kN$ into Equation (51). This produces the equation

$$k = 2 - \frac{8}{N + C} - \frac{2k(N - 1)}{N - 1 + C},$$

or

$$k = \frac{2}{3 + (C - 3)/N} + O(1/N) \quad (52)$$

$$\simeq 2/3.$$

Observe that the slope of the curve for $p = 0.5$ in Figure 4(b) approaches 2/3, in agreement with this analysis.

This result is also in agreement with the theory of branching processes. According to Karp and Pearl [3], the number of nodes in a finite tree produced by a branching process is given by

$$\lim_{N \to \infty} \{\textit{expected number of internal nodes in a finite tree}\}$$

$$= \frac{1}{1 - f'(F_\infty)} \quad \text{if} \quad f'(F_\infty) \neq 1,$$

$$= \frac{N}{3} \quad \text{if} \quad f'(F_\infty) = 1, \quad (53)$$

where $f(s)$ is the generating function from Equation (25). In our case, $f'(F_\infty) = 2p(q + pF_\infty)$. We wish to find $L_F(N)$, the number of branches in the search tree which is exactly twice the number of internal nodes counted by (53). The average number of nodes visited is computed by multiplying this count by F_∞, the probability that the branching tree is finite. Hence,

$$L_F(N) = \left(\frac{2}{1 - 2q}\right)\left(\frac{q^2}{p^2}\right) \quad \text{if} \quad p > 0.5,$$

$$= \frac{2N}{3} \quad \text{if} \quad p = 0.5,$$

$$= \frac{2}{1 - 2p} \quad \text{if} \quad p < 0.5. \quad (54)$$

Compare Equation (54) with Equations (46), (48), and (52). Karp and Pearl do not give a derivation of the formula for the case $p = 0.5$, so we are not sure whether their reasoning is similar to the reasoning applied in this paper.

A major advantage of the approach proposed in this paper is that it can be used to solve Equation (21) for $L_S(F)$, which is a function that is not directly treated by the theory of branching processes. To find the asymptotic behavior of Equation (21) for $p < 0.5$, we set $F(N) = 1$ and $S(N) = 0$, their asymptotic values. So Equation (21) becomes

$$L_S(N) = (p + p^2 + pq)L_S(N - 1)$$

$$= 2pL_S(N - 1),$$

which has as its solution

$$L_S(N) = (2p)^{N-1} L_S(1). \quad (55)$$

This function goes to 0 exponentially in N. For $p > 0.5$, we substitute

$$F(N) = 1 - S(N) = q^2/p^2$$

and

$$L_F(N) = \frac{2q^2}{p^2(1 - 2q)}$$

to obtain, after several algebraic simplifications,

$$L_S(N) = L_S(N - 1) + 1/p.$$

This equation has the solution

$$L_S(N) = L_S(1) + (N - 1)/p. \quad (56)$$

Therefore the value of Equation (21) asymptotically approaches a linear function of depth whose slope varies inversely with the survival probability. This analysis is confirmed by the curves plotted in Figure 4(c).

For $p = 0.5$ we use the same approach used to derive Equation (52). Specifically, we assume that

$$F(N) = 1 - \frac{4}{(N + C)},$$

$$S(N) = \frac{4}{(N + C)},$$

$$L_F(N) = 2N/3,$$

and

$$L_S(N) = kN.$$

After tedious manipulation we discover that $k \simeq 1/3$. Figure 4(c) shows that $L_S(N)$ for $p = 0.5$ is linear, with a slope of 1/3 for $N \geq 10$. The figure shows that the curve does not go through the origin, so that a good approximation must add a constant offset to the linear slope, and thus

$$L_S(N) \simeq N/3 + C \tag{57}$$

for some constant C.

A summary of the formulas derived in this section appears in **Table 1.**

5. Effects of leaf probabilities

We have shown that an algorithm that terminates when it encounters the first leaf has a low average complexity provided that the cutoff-probability distribution is uniform across the tree. But the algorithm in Section 2 does not terminate at its first leaf, and it has an average complexity that could grow exponentially in the depth of the tree. In this section we consider what happens when the leaf nodes have a different probability of cutoff than do the internal nodes of the tree. We discover that the average number of nodes visited does increase in some cases, but the function remains a linear function of the depth.

This model presumes that the probability of success at a leaf is the probability p_0, which is constant for all leaves in the tree and need not be equal to the survival probability of an internal branch. To find the average number of nodes visited, we simply substitute different initial values in Equations (46) and (56). Note that the dependence on the initial value of Equation (46) dies out exponentially for sufficiently deep trees, whereas the initial value of Equation (56) provides a constant offset for the solution. When we evaluate the functions themselves as described in Equations (8), (20), and (21) using p_0 and $q_0 = 1 - p_0$ in place of p and q, we obtain the plots shown in **Figures 7, 8,** and **9.** Figure 7 shows the plots for a survival probability of 0.5. Note that all of the curves for the probability of search failure converge to a common asymptote. Shallow trees behave differently because the leaf nodes are close to the root, and the different probabilities of success at the leaves strongly influence the probability of failure of a search of the tree. But deep trees do not depend strongly on the leaf probabilities, because if a search fails, it is most likely to fail high up in the tree without ever visiting a leaf. The curves for the functions $L_F(N)$ and $L_S(N)$ asymptotically become linear and parallel to the curve for which the leaf probability is equal to the survival probability. The offset of an asymptotic curve from the asymptote for $p = 0.5$ is a function of the leaf-success probability. The plots do not show all of the slopes becoming equal because convergence is very slow for this special case of survival probability. If the curves were extended to show greater (and less realistic) depths, the curves would be parallel straight lines. Figures 8 and 9 show similar curves for higher survival probabilities. We expect to see a greater impact from the leaf-survival probabilities because with high probability the search reaches the leaves of the tree. We can set p_0 to a low value to force failure to occur at a leaf and increase the probability of backtracking once a leaf is reached. Indeed the curves show this effect. But trees with depth greater than 10 to 20 exhibit almost no dependence on the leaf-success probability in the graphs for $F(N)$ and $L_F(N)$. The behavior of the failure-effort function agrees with the analysis above. Similarly, the curves for the success-effort function $L_S(N)$ asymptotically become parallel straight lines. The slope of these lines is inversely proportional to N, which is consistent with our derivations. Consequently, the analyses and the plots confirm that this model of search leads to algorithms that examine a number of nodes that grows at most linearly in the depth of the tree for all values of the model's parameters for sufficiently great tree depths.

We conclude that the model is quite robust since it does not depend strongly on the assumption that cutoff probabilities are uniform throughout a tree, nor does it depend strongly on the search terminating at the first leaf. If over a small neighborhood the cutoff probability q exceeds 0.5, within that neighborhood the search will almost surely cut off, regardless of the probabilities of nodes that lie

Table 1 Summary of formulas: Mean values of search functions.

Case 1: $p < 0.5$

$$F(N) \simeq 1 - \frac{K(p)(2p)^N(1-2p)}{1 - p^2K(p)(2p)^N} \simeq 1 \quad \text{as} \quad N \to \infty$$

where $K(p) = \dfrac{(1-q^2)}{2p(1-2p) + 2p^3(1-q^2)}$

$$L_F(N) \simeq (2p)^{N-1}L_F(1) + 2\left[\frac{1-(2p)^{N-1}}{1-2p}\right] \simeq \frac{2}{1-2p} \quad \text{as} \quad N \to \infty$$

$$L_S(N) \simeq (2p)^{N-1}L_S(1) \simeq 0 \quad \text{as} \quad N \to \infty$$

Case 2: $p = 0.5$

$$F(N) \simeq 1 - \frac{4}{(N+C)} \simeq 1 \quad \text{as} \quad N \to \infty$$

$$L_F(N) \simeq 2N/3 \quad \text{as} \quad N \to \infty$$

$$L_S(N) \simeq N/3 \quad \text{as} \quad N \to \infty$$

Case 3: $p > 0.5$

$$F(N) \simeq \frac{q^2}{p^2} - \frac{K(p)(2q)^N(1-2q)}{1 - p^2K(p)(2q)^N} \simeq \frac{q^2}{p^2} \quad \text{as} \quad N \to \infty$$

where $K(p) = \dfrac{q^2(1+p)}{2p^2(1-2q+q^3+pq^3)}$

$$L_F(N) \simeq (2q)^{N-1}L_F(1) + 2\left[\frac{1-(2q)^{N-1}}{1-2q}\right]\left(\frac{q^2}{p^2}\right)$$

$$\simeq \left(\frac{2}{1-2q}\right)\left(\frac{q^2}{p^2}\right) \quad \text{as} \quad N \to \infty$$

$$L_S(N) \simeq L_S(1) + (N-1)/p \quad \text{as} \quad N \to \infty$$

After tedious manipulation we discover that $k \simeq 1/3$. Figure 4(c) shows that $L_S(N)$ for $p = 0.5$ is linear, with a slope of 1/3 for $N \geq 10$. The figure shows that the curve does not go through the origin, so that a good approximation must add a constant offset to the linear slope, and thus

$$L_S(N) \simeq N/3 + C \tag{57}$$

for some constant C.

A summary of the formulas derived in this section appears in **Table 1.**

5. Effects of leaf probabilities

We have shown that an algorithm that terminates when it encounters the first leaf has a low average complexity provided that the cutoff-probability distribution is uniform across the tree. But the algorithm in Section 2 does not terminate at its first leaf, and it has an average complexity that could grow exponentially in the depth of the tree. In this section we consider what happens when the leaf nodes have a different probability of cutoff than do the internal nodes of the tree. We discover that the average number of nodes visited does increase in some cases, but the function remains a linear function of the depth.

This model presumes that the probability of success at a leaf is the probability p_0, which is constant for all leaves in the tree and need not be equal to the survival probability of an internal branch. To find the average number of nodes visited, we simply substitute different initial values in Equations (46) and (56). Note that the dependence on the initial value of Equation (46) dies out exponentially for sufficiently deep trees, whereas the initial value of Equation (56) provides a constant offset for the solution. When we evaluate the functions themselves as described in Equations (8), (20), and (21) using p_0 and $q_0 = 1 - p_0$ in place of p and q, we obtain the plots shown in **Figures 7, 8,** and **9.** Figure 7 shows the plots for a survival probability of 0.5. Note that all of the curves for the probability of search failure converge to a common asymptote. Shallow trees behave differently because the leaf nodes are close to the root, and the different probabilities of success at the leaves strongly influence the probability of failure of a search of the tree. But deep trees do not depend strongly on the leaf probabilities, because if a search fails, it is most likely to fail high up in the tree without ever visiting a leaf. The curves for the functions $L_F(N)$ and $L_S(N)$ asymptotically become linear and parallel to the curve for which the leaf probability is equal to the survival probability. The offset of an asymptotic curve from the asymptote for $p = 0.5$ is a function of the leaf-success probability. The plots do not show all of the slopes becoming equal because convergence is very slow for this special case of survival probability. If the curves were extended to show greater (and less realistic) depths, the curves would be parallel straight lines. Figures 8 and 9 show similar curves for higher survival probabilities. We expect to see a greater impact from the leaf-survival probabilities because with high probability the search reaches the leaves of the tree. We can set p_0 to a low value to force failure to occur at a leaf and increase the probability of backtracking once a leaf is reached. Indeed the curves show this effect. But trees with depth greater than 10 to 20 exhibit almost no dependence on the leaf-success probability in the graphs for $F(N)$ and $L_F(N)$. The behavior of the failure-effort function agrees with the analysis above. Similarly, the curves for the success-effort function $L_S(N)$ asymptotically become parallel straight lines. The slope of these lines is inversely proportional to N, which is consistent with our derivations. Consequently, the analyses and the plots confirm that this model of search leads to algorithms that examine a number of nodes that grows at most linearly in the depth of the tree for all values of the model's parameters for sufficiently great tree depths.

We conclude that the model is quite robust since it does not depend strongly on the assumption that cutoff probabilities are uniform throughout a tree, nor does it depend strongly on the search terminating at the first leaf. If over a small neighborhood the cutoff probability q exceeds 0.5, within that neighborhood the search will almost surely cut off, regardless of the probabilities of nodes that lie

Table 1 Summary of formulas: Mean values of search functions.

Case 1: $p < 0.5$

$$F(N) \simeq 1 - \frac{K(p)(2p)^N(1-2p)}{1 - p^2K(p)(2p)^N} \simeq 1 \quad \text{as} \quad N \to \infty$$

where $K(p) = \dfrac{(1-q^2)}{2p(1-2p) + 2p^3(1-q^2)}$

$$L_F(N) \simeq (2p)^{N-1}L_F(1) + 2\left[\frac{1-(2p)^{N-1}}{1-2p}\right] \simeq \frac{2}{1-2p} \quad \text{as} \quad N \to \infty$$

$$L_S(N) \simeq (2p)^{N-1}L_S(1) \simeq 0 \quad \text{as} \quad N \to \infty$$

Case 2: $p = 0.5$

$$F(N) \simeq 1 - \frac{4}{(N+C)} \simeq 1 \quad \text{as} \quad N \to \infty$$

$$L_F(N) \simeq 2N/3 \quad \text{as} \quad N \to \infty$$

$$L_S(N) \simeq N/3 \quad \text{as} \quad N \to \infty$$

Case 3: $p > 0.5$

$$F(N) \simeq \frac{q^2}{p^2} - \frac{K(p)(2q)^N(1-2q)}{1 - p^2K(p)(2q)^N} \simeq \frac{q^2}{p^2} \quad \text{as} \quad N \to \infty$$

where $K(p) = \dfrac{q^2(1+p)}{2p^2(1-2q+q^3+pq^3)}$

$$L_F(N) \simeq (2q)^{N-1}L_F(1) + 2\left[\frac{1-(2q)^{N-1}}{1-2q}\right]\left(\frac{q^2}{p^2}\right)$$

$$\simeq \left(\frac{2}{1-2q}\right)\left(\frac{q^2}{p^2}\right) \quad \text{as} \quad N \to \infty$$

$$L_S(N) \simeq L_S(1) + (N-1)/p \quad \text{as} \quad N \to \infty$$

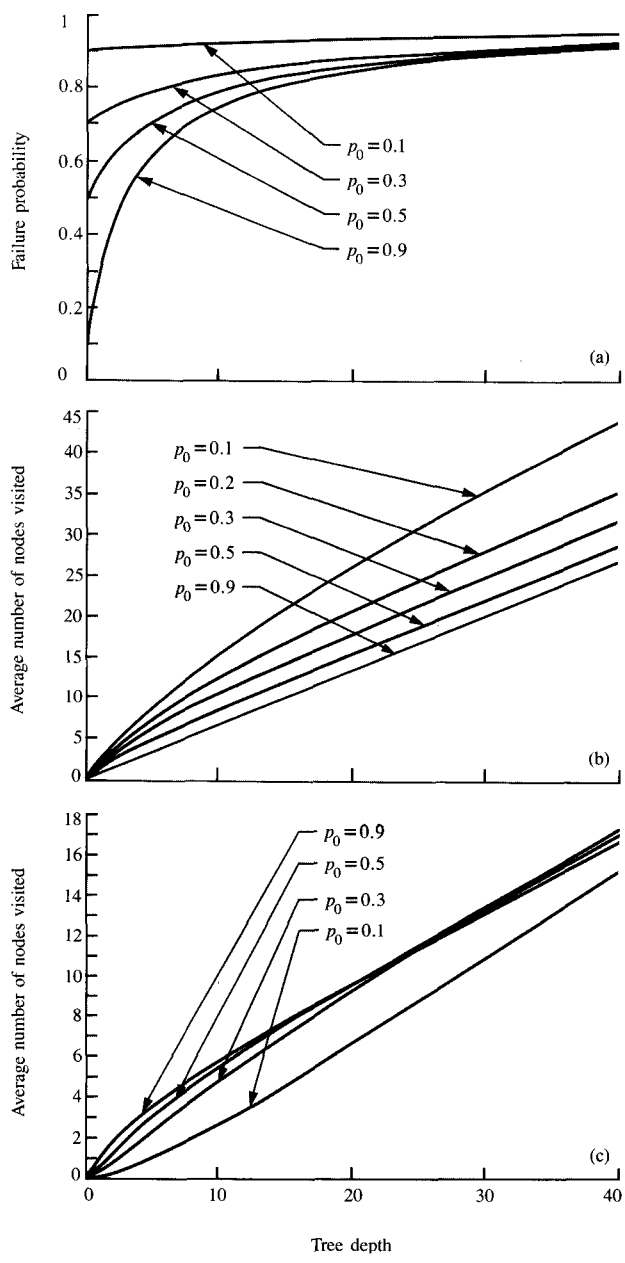

Figure 7

Search statistics for branch-survival probability $p=0.5$. (a) Search-failure probability as a function of leaf-survival probability p_0. (b) Expected number of nodes visited on paths that fail as a function of leaf-survival probability p_0. (c) Expected number of nodes visited on paths that succeed as a function of leaf-survival probability p_0.

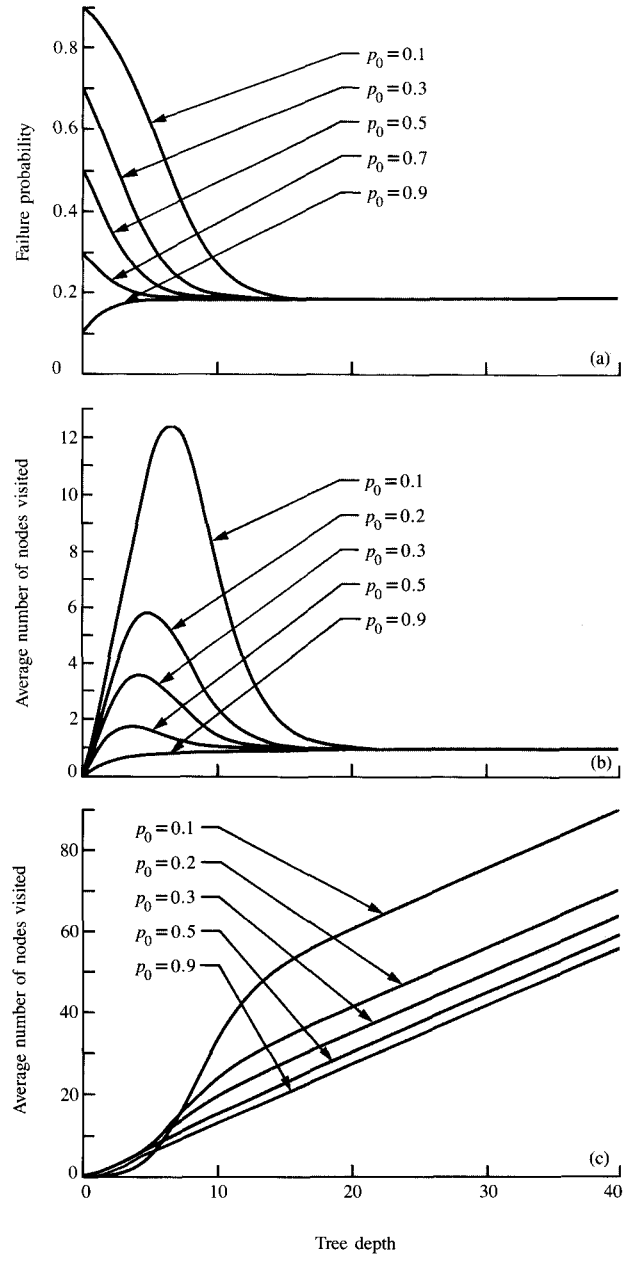

Figure 8

Search statistics for branch-survival probability $p=0.7$. (a) Search-failure probability as a function of leaf-survival probability p_0. (b) Expected number of nodes visited on paths that fail as a function of leaf-survival probability p_0. (c) Expected number of nodes visited on paths that succeed as a function of leaf-survival probability p_0.

beyond the cutoff points. When the search is far enough from regions of the tree that have markedly different cutoff probabilities, the remote regions have little influence on the search. Similarly, if a search reaches a leaf node and does not succeed, it backtracks into the tree and seeks another leaf node. The cost of backtracking and seeking additional leaves adds a cost to the search that is independent of N but does depend on the probability of failing at a leaf node. Hence, by dropping the two assumptions of uniform cutoff probability within the tree and success on encountering the first leaf, we still have a search that visits an average number of nodes that grows linearly in the depth of the tree.

The critical aspect of the search algorithm that leads to this efficient behavior is that there does exist a cutoff probability that can eliminate large sections of the search tree. Therefore a search either fails quickly, or quickly reaches the leaves of the search tree where it examines possible solutions. If the search backtracks to the interior of the search tree, the cutoff probability eliminates unlikely nodes and quickly takes the search to other likely solutions at the leaves of the tree. The search basically does not expend much time in the interior of the tree examining improbable paths.

6. Derivation of variance

This section derives the variance of the averages of the functions defined in Section 4. The objective is to show that the averages are quite meaningful because the variances are very small. In some cases, the variances depend only on p and are independent of the depth of the tree. We use the same method used in Section 4 to solve recurrence equations that describe the variance functions. The details are quite complicated, however, and are generally omitted here. The results have been checked for accuracy with the aid of a symbolic expression analyzer.

Although the method for computing variance is straightforward, the details are rather complicated, so we state only the method and the final results. The basic idea follows the techniques of Section 3, in which we formulate recursion equations for the variance of the failure-effort and success-effort functions. The recurrences depend on the functions $F(N)$, $S(N)$, $L_F(N)$, and $L_S(N)$ defined in Section 3. But because each of these functions converges to an asymptotic function of N, as given in Table 1, we can replace the function with its asymptotic formula and obtain an approximation for the recurrence for the variance formulas. The recurrence obtained by using asymptotic approximations is far simpler than the original recurrence, and it reduces to very simple expressions in most of the cases. With this structure in mind, here is a derivation for the function $V_F(N)$, the variance of the average failure-effort function $L_F(N)$:

$$V_F(N) = q^2 2^2 + 2qp \sum_{i \in FP(N-1)} p_i[2 + length(path_i)]^2 + p^2 \sum_{i \in FP(N-1)} p_i \sum_{j \in FP(N-1)} p_j[2 + length(path_i) + length(path_j)]^2 - [L_F(N)]^2. \quad (58)$$

Equation (58) expresses the variance as the mean square minus the square of the mean, where the mean square is made up of the first three terms of the equation. These three terms correspond to those of Equation (15), except that the path lengths in Equation (58) are the squares of those in Equation (15). We use the same notation in Equation (58) as in Equation (15), so that the summations are taken over all failure paths in a tree of depth $N-1$. To evaluate Equation (58) we square the various terms and perform the

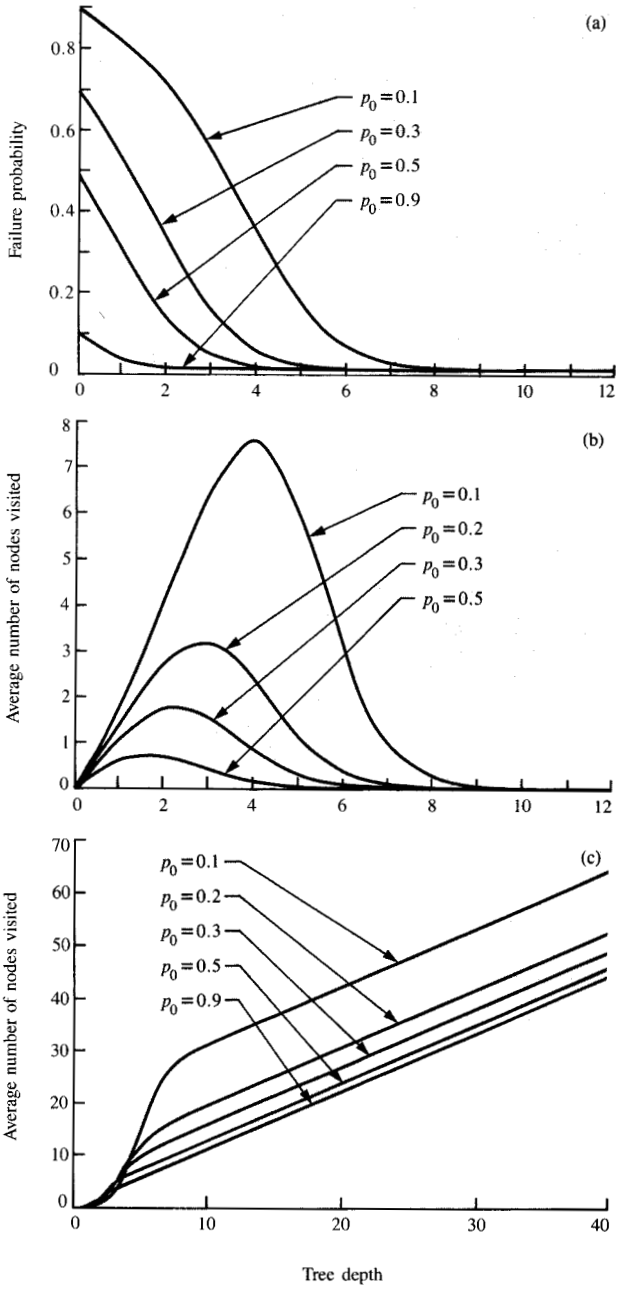

Figure 9

Search statistics for branch-survival probability $p = 0.9$. (a) Search-failure probability as a function of leaf-survival probability p_0. (b) Expected number of nodes visited on paths that fail as a function of leaf-survival probability p_0. (c) Expected number of nodes visited on paths that succeed as a function of leaf-survival probability p_0.

summations, replacing the results as necessary with the functions $F(N-1)$, $L_F(N-1)$, and a new function, $Q_F(N-1)$, defined as the mean square failure path for trees of depth $N-1$. This substitution yields, after some algebraic manipulation,

$$V_F(N) = 2[q + pF(N-1)]\{2q + p[2F(N-1) + 4L_F(N-1) + Q_F(N-1)]\} + 2p^2[L_F(N-1)]^2 - [L_F(N)]^2. \quad (59)$$

Equation (59) can be simplified slightly by using Equation (20) to obtain

$$V_F(N) = 2L_F(N) + 2p[q + pF(N-1)][2L_F(N-1) + Q_F(N-1)] + 2p^2[L_F(N-1)]^2 - [L_F(N)]^2. \quad (60)$$

Since, by definition,

$$V_F(N-1) = Q_F(N-1) - [L_F(N-1)]^2, \quad (61)$$

after substitution of (61) into (60) and some algebraic reduction we have

$$V_F(N) = 2p[q + pF(N-1)]V_F(N-1) + L_F(N)[2 - L_F(N)] + 2pL_F(N-1)\{pL_F(N-1) + [q + pF(N-1)][2 + L_F(N-1)]\}. \quad (62)$$

Equation (62) is in a form that can be evaluated easily when we replace the functions with their asymptotes. From Table 1, for $p < 0.5$ we discover that Equation (62) approaches

$$V_F(N) \simeq 2p(q+p)V_F(N-1) + \frac{2}{1-2p}\left(2 - \frac{2}{1-2p}\right) + 2p\,\frac{2}{1-2p}\left[\frac{2}{1-2p} + (q+p)\left(2 + \frac{2}{1-2p}\right)\right] \simeq 2pV_F(N-1) + \frac{8pq}{(1-2p)^2}. \quad (63)$$

The general form of Equation (63) is essentially the same as

$$V_F(N) = \alpha V_F(N-1) + \beta, \quad (64)$$

where α and β are functions of p but not of N. The solution to Equation (64) is

$$V_F(N) = \alpha^N V_F(0) + \frac{\beta(1-\alpha^N)}{1-\alpha}. \quad (65)$$

For this case, the coefficient $\alpha = 2p < 1$, so the first term goes to 0 exponentially, and the limit of the second term is $\beta/(1-\alpha)$, from which we obtain

$$V_F(N) \simeq \frac{8pq}{(1-2p)^3}. \quad (66)$$

For the case for which $p > 0.5$, we obtain an equation similar to (64) except that

$$\alpha = 2q \quad (67)$$

and

$$\beta = \frac{4q^2[p^2(p-q) + q(2p^3 + 2q^2 - q)]}{p^4(1-2q)^2}. \quad (68)$$

Since $\alpha = 2q < 1$, the first term of the solution in Equation (65) vanishes exponentially in N, and the asymptotic solution is

$$V_F(N) \simeq \frac{\beta}{(1-\alpha)}, \quad (69)$$

where α and β are given by Equations (67) and (68).

The variance for the case for $p = 0.5$ is somewhat more challenging: The analysis is complicated because many terms depend on N in this case, whereas they are independent of N in the other two cases. The analysis technique we propose is to examine Equation (62) with all functional dependencies replaced by their asymptotic values. Specifically we use the following four relations from Table 1:

$$F(N) = 1 - \frac{4}{N},$$

$$S(N) = 1 - F(N) = \frac{4}{N},$$

$$L_F(N) = \frac{2N}{3},$$

and

$$L_S(N) = \frac{N}{3}.$$

When we use these approximations together with $p = q = 0.5$ and substitute into (62), we obtain, after regrouping terms,

$$V_F(N) = \frac{N-3}{N-1}V_F(N-1) + \frac{2N^2}{9} + O(N). \quad (70)$$

The solution to Equation (70) is

$$V_F(N) = \frac{2}{9(N-1)(N-2)}\sum_{i=3}^{N}(i-2)(i-1)i^2 + O(N^2) \simeq \frac{2N^3}{45} + O(N^2). \quad (71)$$

In this derivation, we have concentrated on the dominant terms. The terms of lower order can be derived as well simply by carrying through the tedious detail. Note that the variance grows as the cube of N while the mean grows only linearly with N.

This completes the derivation of the variance of the failure effort for all values of p. Only the case $p = 0.5$ has a variance that depends on the depth of the tree. The variance for the cases for which $p \neq 0.5$ asymptotically approaches a function that depends on p only and not on the depth of the search tree. Consequently, the observed value of $L_F(N)$ in these cases is expected to fall within a bounded region from the mean value where the bound is independent of the depth of the tree.

The derivation of $V_S(N)$, the variance of the success function, follows in a similar fashion. With some algebraic manipulation we can produce the recurrence equation

$$\begin{aligned} V_S(N) &= p[1 + q + pF(N-1)]V_S(N-1) \\ &\quad + p[1 + q + pF(N-1)] \\ &\quad \times \{4S(N-1) + L_S(N-1)[4 + L_S(N-1)]\} \\ &\quad + 2p[pL_F(N-1) - 1][2S(N-1) + L_S(N-1)] \\ &\quad + pS(N-1)\{1 + pV_F(N-1) + p[L_F(N-1)]^2\} \\ &\quad - [L_S(N)]^2. \end{aligned} \tag{72}$$

In spite of the complexity of (72), it simplifies greatly when we supply the asymptotic values from Table 1. For the case $p < 0.5$, since the asymptotic limit of $S(N)$ and $L_S(N)$ is 0, we have

$$\begin{aligned} V_S(N) &\simeq p(1 + q + p)V_S(N-1) \\ &= 2pV_S(N-1). \end{aligned}$$

The solution to this equation is

$$V_S(N) = (2p)^{(N-1)}V_S(1), \tag{73}$$

which goes to 0 exponentially in N.

For the case $p > 0.5$, the equation becomes messy because the terms in the equation do not vanish. Most terms converge to asymptotic functions that depend on p only and not on N. The function $L_S(N)$, however, converges asymptotically to N/p. To solve Equation (73), it is necessary to group the terms of (72) according to whether they depend on $L_S(N)$ or not. When we do this and note that

$$L_S(N-1)^2 - L_S(N)^2 \simeq -\frac{2(N-1)}{p} + \cdots,$$

we find that Equation (72) eventually simplifies to

$$V_S(N) \simeq V_S(N-1) + \frac{2(N-1)q^2}{(1-2q)p^2} + \alpha, \tag{74}$$

where α is a function of p and is independent of N. We can solve Equation (74) by using techniques described above for Equation (70), and we obtain

$$V_S(N) = V_S(1) + \frac{N(N+1)q^2}{(1-2q)p^2} + (N-1)\alpha. \tag{75}$$

Table 2 Summary of formulas: Variance of search functions.

Case 1: $p < 0.5$

$$V_F(N) \simeq \frac{8pq}{(1-2p)^3} \quad \text{as} \quad N \to \infty$$

$$V_S(N) \simeq 0 \quad \text{as} \quad N \to \infty$$

Case 2: $p = 0.5$

$$V_F(N) \simeq \frac{2N^3}{45} \quad \text{as} \quad N \to \infty$$

$$V_S(N) \simeq \frac{7N^3}{180} \quad \text{as} \quad N \to \infty$$

Case 3: $p > 0.5$

$$V_F(N) \simeq \frac{\beta}{(1-2q)}$$

where $\beta = \dfrac{4q^2[p^2(p-q) + q(2p^3 + 2q^2 - q)]}{p^4(p-q)^2}$

$$V_S(N) \simeq \frac{N(N+1)q^2}{(1-2q)p^2} \quad \text{as} \quad N \to \infty$$

For the case $p = 0.5$, after substituting the asymptotic values used to derive Equation (70), we obtain

$$V_S(N) = \frac{N-2}{N-1}V_S(N-1) + \frac{7N^2}{45} + O(N). \tag{76}$$

The solution to Equation (76) is

$$\begin{aligned} V_S(N) &= \frac{7}{45(N-1)}\sum_{i=2}^{N}(i-1)i^2 \\ &\quad + O(N^2) \\ &\simeq \frac{7N^3}{180} + O(N^2). \end{aligned} \tag{77}$$

A summary of the variance formulas appears in **Table 2**.

7. Summary and conclusions

We have shown two models for searching trees. The first model has no internal cutoff and does not terminate at the first leaf. The discussion shows that this model can yield an average search effort that grows only linearly in the depth of the tree if the search is very clever at picking the branch to examine first. If the search is not particularly clever in making this choice, the average search complexity will grow exponentially in the depth of the tree.

The second model is more interesting because it shows a way for building efficient search algorithms for NP-complete problems. The search algorithm is assumed to be able to identify fruitless search paths early and to prune these from

the search tree. It also requires the search to terminate successfully at the first leaf node. We presume that at each node in the tree the search algorithm performs a computation that decides whether to cut off a branch. If the cost of this calculation grows only polynomially in the depth of the tree, the average complexity of the total search will be bounded by a polynomial in the depth of the tree, which is substantially better than the exponential bound for the worst case. We also explored the effect of different leaf-success probabilities on the average search effort, and we discovered that the linear nature of the bounds is not affected by leaf-success probability, although the effort does depend on the probability of success at each leaf. Hence, the fact that the model terminates at the first leaf encountered is not a critical factor in the linear behavior of the complexity functions we analyzed. What appears to be the crucial element of the model is the internal cutoff probability. Many questions remain open about search models for which cutoff probability varies with the depth of the tree, and finding accurate probability models for existing search algorithms that have internal cutoff.

While the results of this paper may have direct influence on some search algorithms, there are many types of search algorithms for which no obvious cutoff computation is available. For example, the Traveling Salesman problem lends itself quite well to various branch-and-bound strategies that take into account the lengths of paths. The numerical measure gives a natural way of ordering partial solutions by their "quality." But what can be done to guide a theorem prover? Are there reasonable functions that can tell a program when to abandon or defer unpromising search paths? If so, then there may be ways to build theorem provers whose average performance is very efficient.

References

1. D. R. Smith, "Random Trees and the Analysis of Branch and Bound Procedures," *J. ACM* **31,** No. 1, 163–188 (January 1984).
2. J. P. Roth, W. G. Bouricius, and P. R. Schneider, "Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits," *IEEE Trans. Electron. Computers* **EC-16,** 567–580 (1967).
3. R. M. Karp and J. Pearl, "Searching for an Optimal Path in a Tree with Random Costs," *Artif. Intell.* **21,** 99–116 (1983).
4. T. Harris, *The Theory of Branching Processes*, Springer-Verlag, Berlin, 1963.
5. T. W. Mullikin, "Branching Processes in Neutron Transport Theory," *Probabilistic Methods in Applied Mathematics*, A. T. Bharucha-Reid, Ed., Academic Press, Inc., New York, 1968.
6. A. V. Aho and N. J. A. Sloane, "Some Doubly Exponential Sequences," *Fibonacci Quart.* **11,** No. 4, 429–437 (1973).
7. J. N. Franklin and S. W. Golomb, "A Function-Theoretic Approach to the Study of Nonlinear Recurring Sequences," *Pacific J. Math.* **56,** No. 2, 455–468 (1975).
8. A. Gottlieb and J. T. Schwartz, "Networks and Algorithms for Very-Large-Scale Parallel Computation," *Computer* **15,** No. 1, 27–36 (1982).
9. C. P. Kruskal and M. Snir, "The Performance of Multistage Interconnection Networks for Multiprocessors," *IEEE Trans. Computers* **C-32,** No. 12, 1091–1098 (December 1983).
10. P. W. Purdom, Jr. and C. A. Brown, *The Analysis of Algorithms*, Holt, Rinehart, and Winston, New York, 1985.
11. D. H. Greene and D. E. Knuth, *Mathematics for the Analysis of Algorithms*, Birkhauser Publishing Co., Boston, 1981.

Received November 13, 1985; accepted for publication January 13, 1986

Harold S. Stone *IBM Research Division, P.O. Box 218, Yorktown Heights, New York 10598.* Dr. Stone is the manager of advanced architecture studies at the IBM Thomas J. Watson Research Center. He has been a faculty member at the University of Massachusetts and Stanford University and has held visiting faculty appointments at institutions throughout the world. Dr. Stone received a Ph.D. in electrical engineering in 1963 from the University of California at Berkeley. His research contributions have been primarily in computer architecture and digital systems design. He is the author, coauthor, or editor of six textbooks. As a consulting editor to Addison-Wesley, McGraw-Hill, and University Microfilms, Dr. Stone has produced four series which contain more than seventy titles in all areas of computer science and engineering. He has been active in both the Association for Computing Machinery and the Institute of Electrical and Electronics Engineers, and has served as Technical Editor of *Computer Magazine* and Governing Board Member of the IEEE Computer Society.

Paolo Sipala *University of Trieste, Department of Electrical Engineering, Via A. Valerio, 10, Trieste, Italy 34127.* Dr. Sipala received a Ph.D. in electronic engineering from the University of Padua, Italy, in 1964. He then joined the Department of Electrical Engineering of the University of Trieste, Italy, where he is currently an Associate Professor. He has spent sabbatical periods in England (University of Newcastle, Imperial College of London) and in the U.S.A. (University of Massachusetts, IBM Thomas J. Watson Research Center). The present paper was conceived and written during his sabbatical with IBM. His main professional interests are in the fields of automata theory and formal languages.