TREE SIZE BY PARTIAL BACKTRACKING

Paul W. Purdom
Computer Science Department
Indiana University
Bloomington, Indiana 47401

TECHNICAL REPORT No. 60

TREE SIZE BY PARTIAL BACKTRACKING

PAUL W. PURDOM

FEBRUARY, 1977

TREE SIZE BY PARTIAL BACKTRACKING

Abstract: Knuth (1) recently showed how to estimate the size of a backtrack tree by repeatedly following random paths from the root. Often the efficiency of his method can be greatly improved by occasionally following more than one path from a node. This results in estimating the size of the backtrack tree by doing a very abbreviated partial backtrack search. An analysis shows that this modification results in an improvement which increases exponentially with the height of the tree. Experimental results for a particular tree of height 84 show an order of magnitude improvement. The measuring method is easy to add to a backtrack program.

1. Introduction

The author had been trying to develop an improved method for multiplying 3 by 3 matrices using a backtrack program when he heard of Knuth's (1) method of estimating the efficiency of backtrack programs. The author had planned to make extensive use of Knuth's method, but found that for his particular trees a huge number of runs were required to obtain even an order of magnitude estimate of the tree sizes. This experience was quite different from that of several others (1,2). The author finally determined that the important characteristic of his trees was their great height combined with the large fraction of nodes on each level which had no sons. Eventually the author discovered a slight modification of Knuth's algorithm which produces a large increase in efficiency.

Knuth (1) gives a good discussion of the uses of backtrack programs, which will not be repeated here. The method he proposes for measuring the size of a backtrack tree is to repeatedly follow random paths from the root and to estimate that the number of nodes in the tree is the average over several runs of $1+d_1 + d_1 d_2 + \ldots$ where d_i is the number of successors to the node on level i. This gives an unbiased estimate of the number of nodes in the tree. It is particularly efficient when used to estimate the size of a complete binary tree of height h, since it concludes that the size is 2^h-1 after examining 2h-1 nodes. On the other hand it has trouble with a tree in which each node has zero or two sons, and no more than one node per level has sons which have sons themselves. Part of such a tree is shown in figure 1, where only the nodes which Knuth's algorithm considers expanding (i.e., nodes with sons) are shown.

The original tree can be obtained by adding two sons to each node with no sons. For such a tree Knuth's algorithm will examine an average of $7-2^{-h+3}$ nodes. While the expected value of the estimate is 4h-1, the most frequent result of a single run will be 7. Knuth's algorithm would have to be run 2^{h-1} times to have a reasonable chance of examining the bottom level. The best way to find the size of a tree of the type shown in figure 1 is to examine all the nodes, using a complete backtrack algorithm.

Tall skinny trees can thus best be measured by doing a complete backtrack search, while short fat trees can be efficiently measured using Knuth's algorithm. For intermediate trees neither of these methods works well. The tree can have so many nodes that it is not practical to do a complete backtrack search, while at the same time it has so many nodes with no successors at each level that Knuth's algorithm has difficulty learning about the deeper levels of the tree. While Knuth (1) offers several suggestions that help in this situation, they are often difficult to apply, and none of them get to the heart of the problem. What is needed is an algorithm that will sample the deeper levels, which usually contain most of the nodes, with about the same probability as the upper levels. The way to increase the number of nodes observed at the deeper levels is to occasionally investigate more than one branch out of a node. If the average number of branches to follow is selected properly then one may be able to look at an average of about one node per level. Doing this will greatly improve the efficiency of the estimating program.

2. Backtrack Measuring Algorithms

A general backtrack program finds all the vectors $(x_1,x_2,\ldots,x_n)$ which satisfy some property $P_n(x_1, x_2,\ldots,x_n)$. There are also intermediate properties $P_k(x_1, x_2,\ldots,x_k)$ such that $P_{k+1}(x_1,x_2,\ldots,x_{k+1}) \supset P_k(x_1\ldots x_k)$ for $o \le k < n$. When $P_k(x_1,x_2,\ldots,x_k)$ is false, backtracking saves considerable work since no extension of $(x_1,x_2,\ldots,x_k)$ needs to be considered as a possible solution.

The following algorithm is the common backtrack algorithm with modifications for measuring tree sizes. The values x_k are kept on a stack, where k is the stack pointer. The parameter n is the length of the vectors $(x_1,x_2,\ldots,x_n)$ which satisfy the final property $P_n(x_1,x_2,\ldots,x_n)$. The function $C(x_1,x_2,\ldots,x_k)$ gives the cost of examining node $(x_1,x_2,\ldots,x_k)$. Setting C to one for all nodes results in a total cost equal to the number of nodes in the tree. The stack entry r_k contains the number of sons remaining to be selected from the node on level k-1. The variable t accumulates the estimated cost of the tree. To monitor the performance of the algorithm the optional arrays c_k and f_k accumulate the estimated cost of examining the nodes on level k and the number of nodes investigated on level k. For convenience the algorithm assumes that the values for x_k are consecutive integers starting with one. Only minor changes are needed to handle arbitrary discrete x_k.

Basic Backtrack Estimating Algorithm

Step B1. [Initialize] Set k to 0, t to 0, and d_0 to 1. For $1 \le j \le n$ set c_j to 0 and f_j to 0.

Step B2. [Go down] If k=n then output the solution $(x_1,x_2,\ldots,x_n)$ and go to step B6. Otherwise set k to k+1. Set a to the number of

values that x_k can take on. If a is zero then go to step B6. Set m to number of values that will be investigated. This must be an integer such that $1 \le m \le a$. Set d_k to d_{k-1} a/m, r_k to m, and x_k to a+1.

Step B3. [More] If x_k is 1 then go to step B6.

Step B4. [Next] Set x_k to x_k-1. With probability $1-r_k/x_k$ reject x_k by going to step B3. Otherwise set r_k to r_k-1, t to $t+d_kC(x_1, x_2, \ldots, x_k)$, c_k to $c_k+d_kC(x_1, x_2, \ldots, x_k)$, and f_k to f_k+1.

Step B5. [Test] If $P_k(x_1,x_2,\ldots, x_k)$ is true then go to step B2.

Step B6. [Go up] Set k to k-1. If k=0 then stop. Otherwise go to step B3.

This algorithm is the traditional backtrack algorithm with additions at steps B1, B2, and B4 for measuring tree size. Step B4 has the algorithm for selecting m of a values at random (3,4). The efficiency of the algorithm depends on the value of m selected at step B2. How to select m will be considered in detail later. However m is selected, the expected value of the sum over all nodes visited on level k of $d_kC(x_1,x_2,\ldots,x_k)$ is the sum of the cost of the nodes on level k.

Knuth's algorithm for measuring tree size modifies the standard backtrack algorithm by first testing all the successors to a node before selecting which one to follow. The following algorithm combines this idea with partial backtracking. It replaces the stack r_k with a stack of sets S_k. The stack S_k contains the set of values remaining to be considered for x_k. The set Q contains all the values of x_k for which $P(x_1,x_2,\ldots,x_k)$ is true.

Modified Backtrack Estimating Algorithm

Step M1. [Initialize] Set k to 0, t to 0, and d_0 to 1. For $1 \le k \le n$ set c_k to 0 and f_k to 0.

Step M2. [Go down] If k=n output solution $(x_1, x_2, \ldots, x_n)$ and go to step M5. Set k to k+1. Set Q to the set of all x_k such that $P_k(x_1, x_2, \ldots, x_k)$ is true. For each x_k tested for inclusion in Q, set t to $t + d_{k-1} C(x_1, x_2, \ldots, x_k)$, and f_k to $f_k + 1$. Set a to the number of elements in Q. If Q is empty then go to step M5. Set m to the number of values that will be investigated, so that $1 \le m \le a$. Set d_k to $d_{k-1} a/m$.

Step M3. [Select values] Set S_k to a randomly selected subset of m values from Q.

Step M4. [Next] If S_k is empty then go to step M5. Otherwise remove an element from S_k and set x_k to the element. Go to step M2.

Step M5. [Go up] Set k to k-1. If k=0 then stop. Otherwise go to step M4.

The stack of sets S_k can be replaced by a stack of values if one is willing to test each node twice. Simplifications can also be made if $m \le 2$ at all times. The modified algorithm does not test P_0, which is nearly always true.

The modified algorithm looks at all the successors of a node, x_{k-1}, to see where $P_k(x_1, x_2, \ldots, x_k)$ is true, but follows only m of them during partial backtracking. If $P_k(x_1, x_2, \ldots, x_k)$ is always false when $(x_1, x_2, \ldots, x_k)$ has no successors (and is not a solution) then the tree of nodes considered for expansion by the modified algorithm can be obtained from the original backtrack tree by deleting the nonsolution nodes which have no successors. Usually

the resulting tree will also have some nodes with no successors. Knuth's (1) algorithm is the modified algorithm with m always equal to one.

The efficiency of either version of the algorithm depends on how m is chosen. If it is always one, then one has essentially Knuth's original algorithm. If it is always set to a then one has complete backtracking. Small values of m make it difficult to observe deep levels in the tree, while large values of m require that a huge number of nodes be examined. There is often an intermediate value of m which will largely avoid the first problem without causing the second.

3. Analysis

Figure 2 shows the first three members of a sequence of sets of trees which are useful for studying the effect of the value of m. For each tree in a set, there is a probability indicating how often the tree is selected from the set. The set T_0 has the tree 0, consisting of just a root, with probability one. For $i > 1$ set T_i has tree 0 with probability 1-b. It also has each tree of the form (t_1,t_2), which has a root, left subtree t_1, and right subtree t_2, where t_1 and t_2 are selected from T_{i-1}. The tree (t_1,t_2) occurs with probability $bp(t_1)p(t_2)$ where $p(t_1)$ is the probability that t_1 is selected from T_{i-1} and $p(t_2)$ is the probability that t_2 is selected from T_{i-1}. The parameter b controls the average size of the trees in T_i.

In the following the efficiency of a tree size estimating program will be measured by the product of the expected number of nodes examined (averaged over all trees in T_i) times the expected variance of the estimate of tree size (averaged over all trees in T_i).

Since this is a measurement of work times error, low numbers indicate high efficiency. This quantity was selected because it is easy to calculate. Although other measures of efficiency might be better, the method of selecting m should apply with little change even with slightly different measures.

Let $p_{it}(n)$ be the probability that tree t has n nodes and that tree t is selected from T_i. Then $p_{it}(n)$ obeys the recurrence

$$p_{10}(n) = \delta_{n1},$$

$$p_{i0}(n) = (1-b)\delta_{n1} \text{ for } i>1, \text{ and}$$

$$p_{i(t_1,t_2)}(n) = b\sum_m p_{i-1,t_1}(m)p_{i-1,t_2}(n-1-m) \text{ for } i>1.$$

The expected number of nodes for a tree in T_i is

$$n_i = \sum_{n,t} np_{it}(n) = 1+2bn_{i-1} = \frac{(2b)^i-1}{2b-1}.$$

The variance is given by

$$v_i = \sum_{n,t} n^2 p_{it}(n) - n_i^2 = 1+8bn_{i-1}+2bv_{i-1}$$

$$= \frac{2(1-b)}{(2b-1)^3}\left[-2b-(2b-1)(2i-1)(2b)^i+(2b)^{2i}\right]$$

For most values of b and i the standard deviation is the same is the same order of magnitude as the average.

When the basic algorithm selects m to be 1 with probability 1-p and to be 2 with probability p, it estimates the size of binary trees according to the recurrences

$$E(0) = 1,$$

$$E(t_1,t_2) = 1+\begin{cases} 2E(t_1) & \text{with probability } (1-p)/2 \\ 2E(t_2) & \text{with probability } (1-p)/2 \\ E(t_1)+E(t_2) & \text{with probability } p. \end{cases}$$

Let $p_{it}(\ell)$ be the probability that ℓ nodes are examined when estimating the size of t and that tree t is selected from T_i. Then

$$p_{i0}(\ell) = (1-b)\delta_{\ell 1} \text{ and}$$

$$p_{(t_1,t_2)}(\ell) = \frac{b}{2}(1-p)\Big[p_{i-1,t_1}(\ell-1)\sum_m p_{i-1,t_2}(m)+p_{i-1,t_2}(\ell-1)\sum_m p_{i-1,t_1}(m)$$

$$+bp\sum_m p_{i-1,t_1}(m)p_{i-1,t_2}(\ell-1-m) \text{ for } i>1.\Big]$$

The expected number of nodes examined (averaging over all trees in T_i and repeated runs of the algorithm) is

$$\ell_i = \sum_{\ell,t} \ell p_{it}(\ell) = 1+b(1+p)\ell_{i-1} = \frac{[b(1+p)]^i-1}{b(1+p)-1} .$$

Let $p_{it}(e)$ be the probability that e is the estimate of the size of tree t and that tree t is selected from T_i. Then

$$p_{i0}(e) = (1-b)\delta_{e1} \text{ and}$$

$$p_{i(t_1,t_2)} = \frac{b}{2}(1-p)\Big[p_{i-1,t_1}\left(\frac{e-1}{2}\right)\sum_m p_{i-1,t_2}(m)+p_{i-1,t_2}\left(\frac{e-1}{2}\right)\sum_m p_{i-1,t_1}(m)\Big]$$

$$+pb\sum_m p_{i-1,t_1}(m)p_{i-1,t_2}(e-1-m)$$

The expected value of the estimate is

$$e_i = \sum_{e,t} e p_{it}(e) = 1+2be_{i-1} = \frac{(2b)^i-1}{2b-1} .$$

The expected value of the square of the estimate is

$$s_i = \sum_{e,t} e^2 p_{it}(e) = 1+2b(2e_{i-1}+pe_{i-1}^2)+2b(2-p)s_{i-1}$$

$$= \frac{1}{(2b-1)^2}\left[\frac{4b^4-2bp-1}{4b-2bp-1} + \frac{2(3-p)(1-b)(2b-1)^2}{(2b-2p-1)(p-1)(2b+p-2)}\,[2b(2-p)]^i\right.$$

$$\left. + \frac{2[2b-p-1]}{p-1}\,(2b)^i + \frac{p(4b^2)^i}{2b+p-2}\right] .$$

the total variance can be calculated from s_i. Of more interest, however, is the internal variance, which measures the expected

variation of the estimate for the size of a tree about the expected value of the estimate of the size of that tree. The internal variance is equal to the total variance minus the variance in the size of the various trees about the expected tree size. Furthermore, variance in the size of the trees is independent of p while the internal variance is zero for p=1. Therefore, the internal variance is

$$v_i = s_i - s_i(p=1)$$

$$= \frac{2(1-b)}{(2b-1)^3}\left[\frac{4b^2(1-p)}{4b-2bp-1} + (2b-1)\left[\frac{1+p}{1-p} + 2i\right](2b)^i + \frac{1-p}{2b+p-2}(4b^2)^i\right.$$

$$\left. + \frac{(2b-1)^3}{(4b-2bp-1)(1-p)(2b+p-2)}\left[2b(2-p)\right]^i\right].$$

The efficiency of the algorithm is indicated by $\ell_i\ v_i$, where a low value indicates high efficiency. For $\frac{1}{2} \le b \le 1$ and i large there are three values of p of interest. When p is small the term that grows like $[2b(2-p)]^i$ dominates the variance. The size of this term decreases with increasing p. The number of nodes examined is small if $b(1+p) \le 1$. Therefore, $p=b^{-1}-1$ is one value of interest. The $[2b(2-p)]^i$ term continues to dominate as long as $4b^2 \le 2b(2-p)$. Therefore, $p=b-2b$ is a second value of interest. Finally, for p=1 the variance is zero. For $b^{-1}-1 \le p \le 2-2b$ both extreme values for p produce local minima in $\ell_i\ v_i$, provided that one is above $\frac{1}{2}$ and the other is below $\frac{1}{2}$. If both extreme values are on the same side of $\frac{1}{2}$, then only the more distant one produces a local minimum. For $\frac{1}{2} \le b < \frac{1}{2}\sqrt{2}$, $p = 2-b$ produces the lower value, while for $\frac{1}{2}\sqrt{2} < b \le 1$, $p = b^{-1}-1$ produces the lower value. For $\frac{2}{3} \le b \le \frac{1}{2}\sqrt{2}$ one may want to set p to $b^{-1}-1$ to avoid looking at the larger number of nodes required when $p = 2-2b$, just as one usually does not use $p = 1$ because it requires looking at a prohibitive number of nodes.

A constant value of p is often not best. Rather than making many runs with $p=b^{-1}-1$ or $2-2b$, it is better to set $p=1$ for several consecutive levels starting at the root and to set $p=b^{-1}-1$ or $2-2b$ for the remaining levels. In practice one would want to make at least 3 runs since it is also important to know the variance of the estimate.

Now consider the modified algorithm. Let $p_{it}(\ell)$ be the probability that ℓ nodes are examined when estimating the size of tree t and that tree t is selected from T_i. Then

$$p_{i0}(\ell) = (1-b)\delta_{\ell 1},\ p_{i(0,0)}(\ell) = b(1-b)^2\delta_{\ell 3},$$

$$p_{i(t_1,0)}(\ell) = b(1-b)p_{i-1,t_1}(\ell-2) \text{ for } t_1\neq 0,$$

$$p_{i(0,t_2)}(\ell) = b(1-b)p_{i-1,t_2}(\ell-2) \text{ for } t_2 \neq 0, \text{ and}$$

$$p_{i(t_1,t_2)}(\ell) = \frac{b}{2}(1-p)\left[p_{i-1,t_1}(\ell-2)\sum_m p_{i-1,t_2}(m)+p_{i-1,t_2}(\ell-2)\sum_m p_{i-1,t_1}(m)\right]$$
$$+ bp\sum_m p_{i-1,t_1}(m)p_{i-1,t_2}(\ell-1-m) \text{ for } t_1 \neq 0 \text{ and } t_2 \neq 0.$$

The expected number of nodes examined is

$\ell_i = \sum_{\ell,t}\ell p_{it}$. Therefore $\ell_1 = 1$, $\ell_2 = 1+2b$, and for $i>2$

$$\ell_i = 1+b^2(1+p)+(2b-b^2(1-p))\ell_{i-1}$$

$$= \frac{1+b^2(1-p)-2b[b(2-b(1-p))]^{i-1}}{1-2b+b^2(1-p)}$$

Let $p_{it}(e)$ be the probability that the modified method estimates that tree t is of size e and that tree t is selected from T_i. Then

$$p_{i0}(e) = (1-b)\delta_{e1},\ p_{i(0,0)}(e) = b(1-b)^2\delta_{e3},$$

$$p_{i(t_1,0)}(e) = b(1-b)p_{i-1,t_1}(e-2) \text{ for } t_1 \neq 0$$

$$p_{i(0,t_2)}(e) = b(1-b)p_{i-1,t_2}(e-2) \text{ for } t_2 \neq 0, \text{ and}$$

$$p_{i(t_1,t_2)}(e) = \frac{b}{2}(1-p)\left[p_{i-1,t_1}\binom{e-1}{2}\sum_m p_{i-1,t_2}(m) + p_{i-1,t_2}\binom{e-1}{2}\sum_m p_{i-1,t_1}(m)\right]$$

$$+ bp\sum_m p_{i-1,t_1}(m)p_{i-1,t_2}(e-1-m) \text{ for } t_1 \neq 0 \text{ and } t_2 \neq 0.$$

The expected value of the estimate is

$e_i = \sum_{e,t} e p_{it}(e)$. Therefore $e_1 = 1, e_2 = 1+2b$,
and for $i>2$, $e_i = 1+2be_{i-1} = \frac{(2b)^i - 1}{2b-1}$.

The expected value of the square of the estimate is

$s_i = \sum_{e,t} e^2 p_{it}(e)$. Therefore $s_1 = 1$, $s_2 = 1+8b$, and for $i>2$

$$s_i = 1-2b(1-p) + 2b^2(1-p) + 4b[2-p-b(1-p)]e_{i-1} + 2bpe_{i-1}^2$$

$$+ 2b(1+b(1-p))s_{i-1}$$

$$= \frac{1}{(2b-1)^3}\left[\frac{1+2b-2b^2(3-p)-8b^2(1-b)(1-p)}{1-2b-2b^2(1-p)}\right.$$

$$+ \frac{4(2b-1)^2(b-1)(b-bp+2)}{(1-2b-2b^2(1-p))(1-b(1+p))(1-p)}[2b(1+b(1-p))]^{i-1}$$

$$\left. + \frac{4[2-b(5-3p) + 2b^2(1-p)]}{1-p}(2b)^{i-1} - \frac{4b^2p}{1-b(1+p)}(4b^2)^{i-1}\right].$$

The expected internal variance is

$v_i = s_i - s_i(p=1)$. For $i>2$

$$v_i = \frac{4(1-b)}{(2b-1)^3}\left[\frac{4b^4(1-p)}{-1+2b+2b^2(1-p)} + (2b-1)\left[\frac{2-3b(1-p)}{1-p} - 2ib\right](2b)^{i-1}\right.$$

$$\left. - \frac{b^2(1-p)}{1-b(1+p)}(4b^2)^{i-1} + \frac{(2b-1)^3(2+b-bp)}{(-1+2b+2b^2(1-p))(1-b(1+p))(1-p)}[2b+1+b(1-p))]^{i-1}\right].$$

Again the efficiency is indicated by $\ell_i v_i$. For $\frac{1}{2} \leq b \leq 1$ and large i there are three values of p of interest. These are $p = (b^{-1}-1)^2$, $p = b^{-1}-1$, and $p = 1$. For $(b^{-1}-1)^2 \leq p \leq b^{-1}-1$ both endpoints produce local minima if b is near $\frac{1}{2}\sqrt{2}$. For

$\frac{1}{2} \leq b < \frac{1}{2}\sqrt{2}$, $p = b^{-1}-1$ produces the lower local minimum whereas for $\frac{1}{2}\sqrt{2} < b \leq 1$, $p = (b^{-1}-1)^2$ produces the lower local minimum. The basic and modified algorithms have about the same efficiency when p is set to the best value for each. The dominant exponential terms have the same base and exponent, and the coefficients do not differ greatly in value. The analysis indicates that the modified algorithm is favored for b > .7325, and the basic algorithm for b < .7325, but this conclusion is dependent on detailed assumptions made in the analysis. Also, the basic algorithm is easier to program.

4. Experimental Results

To test the practical application of these methods, a number of measurements were made. The first set was made on a backtrack program that found ways to multiply 2 by 2 matrices with 7 multiplications. The program was not very sophisticated and it produced a 106,283,567 node binary tree of height 84. Level 48 had the largest number of nodes (16,077,754). The results of these runs are given in tables 1 and 2. Figures 3 and 4 show the efficiency of each method as a function of p, the probability that both branches of the binary tree are investigated. Figures 3 and 4 also have a least square fit of the theory for T_i trees to the data. The value of chi-square is 98.7 for figure 3 and 21.8 for figure 4. The large values of chi-square are probably caused by using measured standard deviations rather than exact values. For comparison figure 3 also has the curve for the parameters that give the best fit to the data in figure 4. The occasional large standard deviations result from the non-gaussian

nature of the distributions generated by the estimating process. The experimental results show that values of $p \neq 0$ can produce a considerable improvement in the efficiency of the estimation process. They also show that the theory for T_i trees provides a practical guide for selecting p.

The second set of runs was made with a slightly improved program which looked for methods to multiply 3 by 3 matrices using 22 multiplications. This program generated a gigantic tree of height 594 with about 10^{42} nodes. The level with the largest number of nodes is near level 300, which has 4×10^{41} nodes. By carefully selecting a value of p for each level it was possible to estimate a size of 7.14×10^{42} with a standard deviation of 4.79×10^{42}. This required examining 7.8×10^{8} nodes. A single value of p was unsuitable because of the variation in the best value at different levels in the tree. With previous methods it would have been impossible to obtain even a rough estimate of the size of this tree.

The experimental data was collected on a TI980 minicomputer, which could test about 10^{8} nodes per day.

Partial backtracking results in an exponential improvement in Knuth's algorithm for estimating tree size. The effect is particularly important for trees with a lot of dead-end branches (b near $\frac{1}{2}$ in the analysis) and for tall trees. The analysis suggests that good results can be obtained by choosing the number of branches to investigate so that each level, from the root down to the levels with the bulk of the nodes, is examined at about the same frequency. Further improvement is obtained by looking at all branches near the root, although one is usually limited by

the number of nodes one can afford to look at.

Acknowledgement: The author wishes to thank Cynthia Brown and David Seaman who developed programs which led to this work.

References

1. Knuth, D. E. "Estimating the Efficiency of Backtrack Programs," Math. of Comp. 1975 (29), pp. 121-236.

2. Bitner, J. R. and E. M. Reingold. "Backtrack Programming Techniques," C.A.C.M. 1975 (18), pp. 651-655.

3. Fan, C. T., M. E. Muller and Ivan Rezucha. "Development of Sampling Plans by Using Sequential (Item by Item) Selecting Techniques and Digital Computers," J. A. Stat. A. 1962 (57), pp. 387-402.

4. Jones, T. G. "A Note on Sampling a Tape-File," C.A.C.M. 1962 (5), p. 343.

Table 1

p (x1/16)	runs	nodes examined ($\times 10^6$)	estimate ($\times 10^8$)	nodes x variance ($\times 10^{21}$)
0	4.89×10^7	2.000	0.56 ±0.18	68±26
1	3.97×10^7	2.000	0.80 ±0.30	176±161
2	3.15×10^7	2.000	1.49 ±0.74	1099±1086
3	2.39×10^7	2.000	0.757±0.055	6.02±0.66
4	1.66×10^7	2.000	0.998±0.072	10.4 ±5.5
5	9.69×10^6	2.000	0.991±0.027	1.49±0.22
6	4.29×10^6	2.000	1.058±0.036	2.65±1.13
7	1.39×10^6	2.000	1.064±0.030	1.83±0.82
8	548355	3.000	1.064±0.019	1.16±0.46
9	85854	2.000	1.047±0.018	0.70±0.11
10	31541	3.322	1.049±0.016	0.86±0.08
11	5959	2.905	1.029±0.026	1.95±0.50
12	860	2.005	1.128±0.057	6.4 ±1.2
13	195	2.009	1.035±0.074	11.0 ±2.4
14	40	2.045	1.09 ±0.10	21.1 ±5.7
15	10	2.389	1.21 ±0.22	119 ±36
16	1	1.063	1.06283576	0

Results of the basic method with various values of p. Numbers after the ± signs are standard deviations. Not all runs were useful for obtaining the standard errors for the last column.

Table 2

p (x1/16)	runs	nodes examined ($\times 10^6$)	estimate ($\times 10^8$)	nodes x variance ($\times 10^{21}$)
0	3.85×10^6	2.000	0.999±0.030	1.76±0.60
1	2.63×10^6	2.000	1.046±0.021	0.87±0.11
2	1.61×10^6	2.000	1.064±0.014	0.40±0.28
3	860153	2.000	1.123±0.053	5.6 ±4.9
4	397856	2.000	1.064±0.016	0.49±0.11
5	161623	2.000	1.060±0.019	0.76±0.30
6	60744	2.000	1.077±0.015	0.47±0.04
7	21507	2.000	1.055±0.019	0.73±0.18
8	3778	1.000	1.094±0.058	3.4 ±2.2
9	1265	1.001	1.083±0.044	1.97±0.25
10	444	1.005	0.99 ±0.13	4.5 ±1.3
11	150	1.004	1.044±0.075	6.0 ±1.6
12	64	1.021	0.93 ±0.13	17.6 ±7.3
13	21	1.018	1.05 ±014	19.8 ±4.7
14	9	1.354	0.58 ±0.38	199 ±48
15	5	1.833	1.04 ±0.15	89 ±48

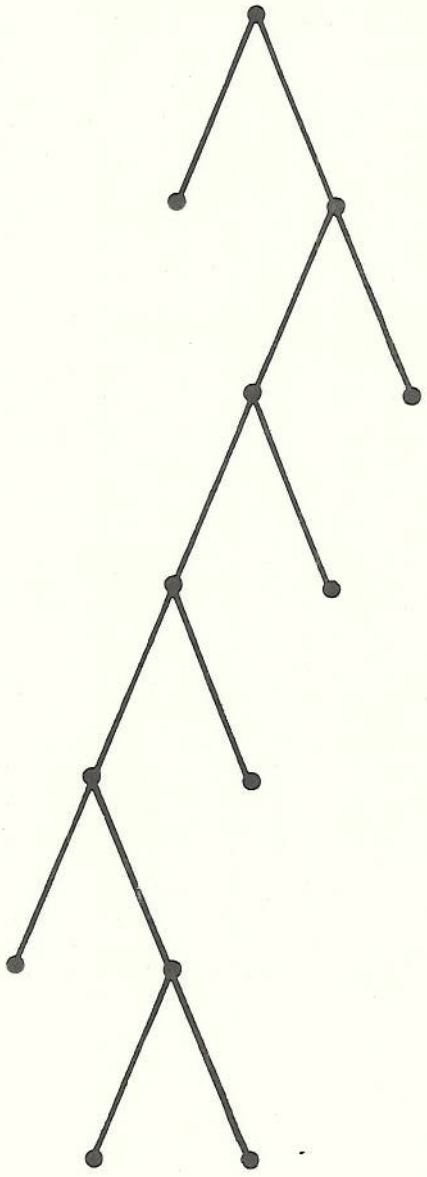
Results of the modified method with various values of p. Not all runs were useful for obtaining the standard errors for the last column.

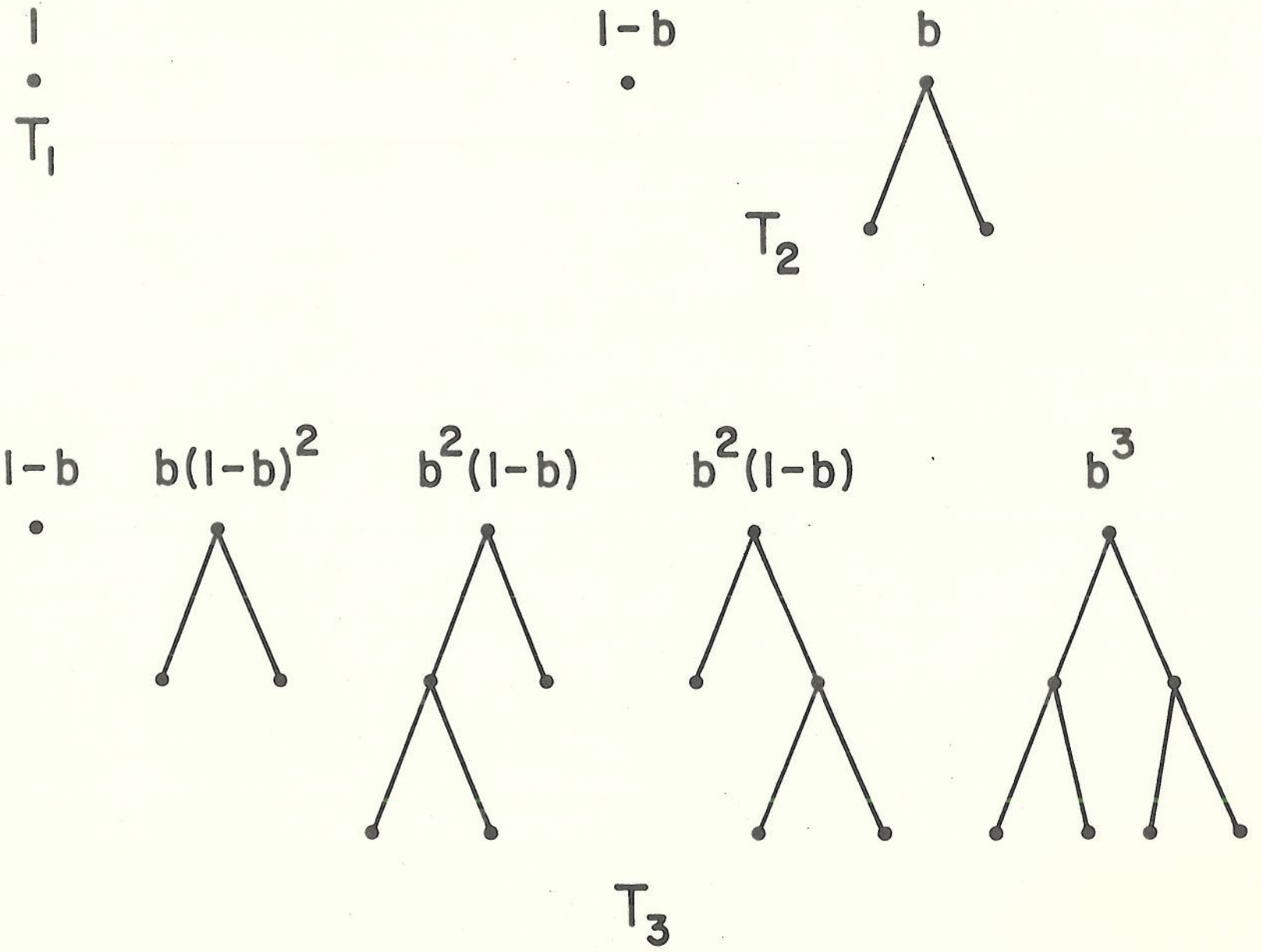
Figure 1. A tall skinny backtrack tree.

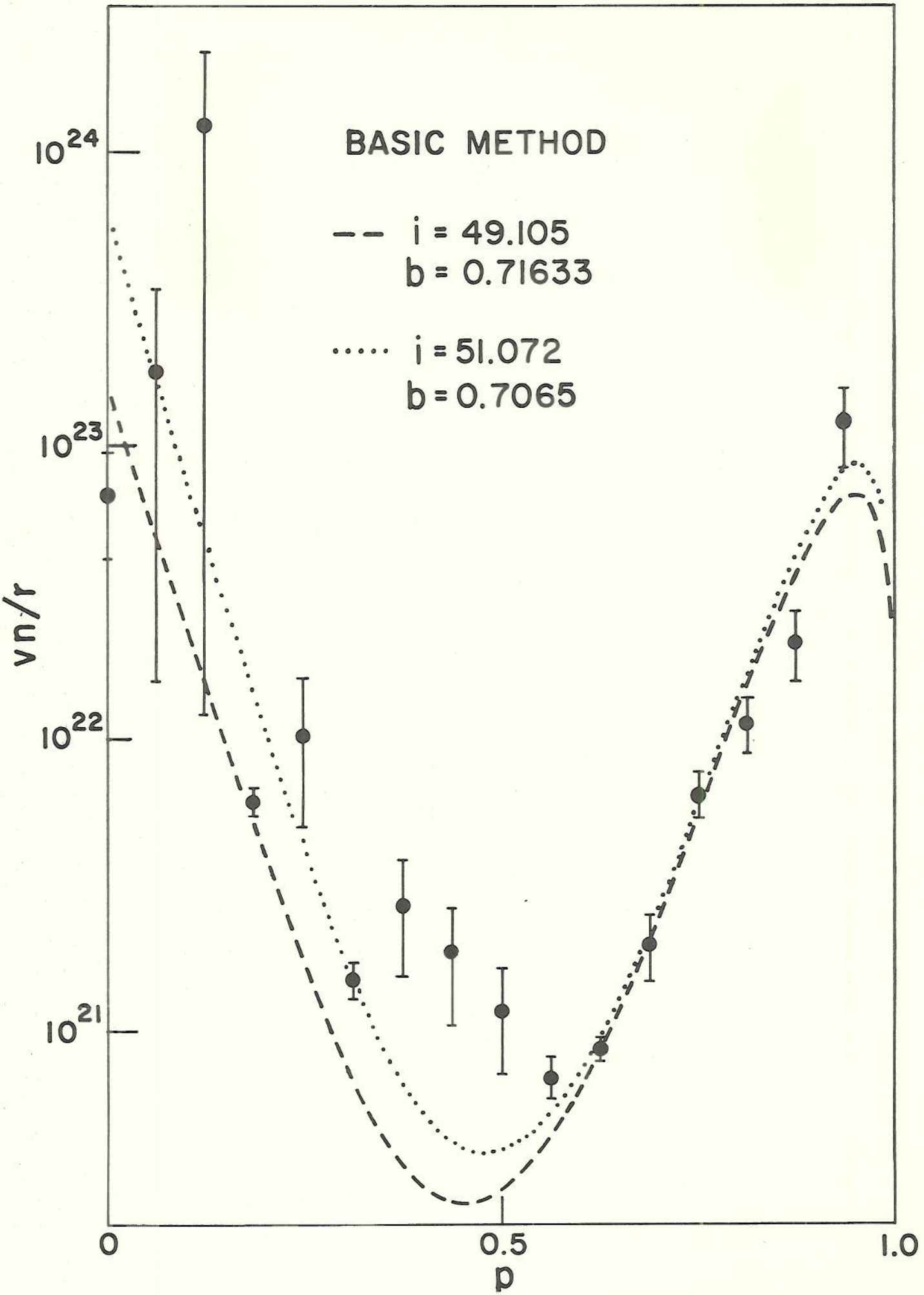
Figure 2. The trees in T_1, T_2, and T_3 with their associated probabilities.

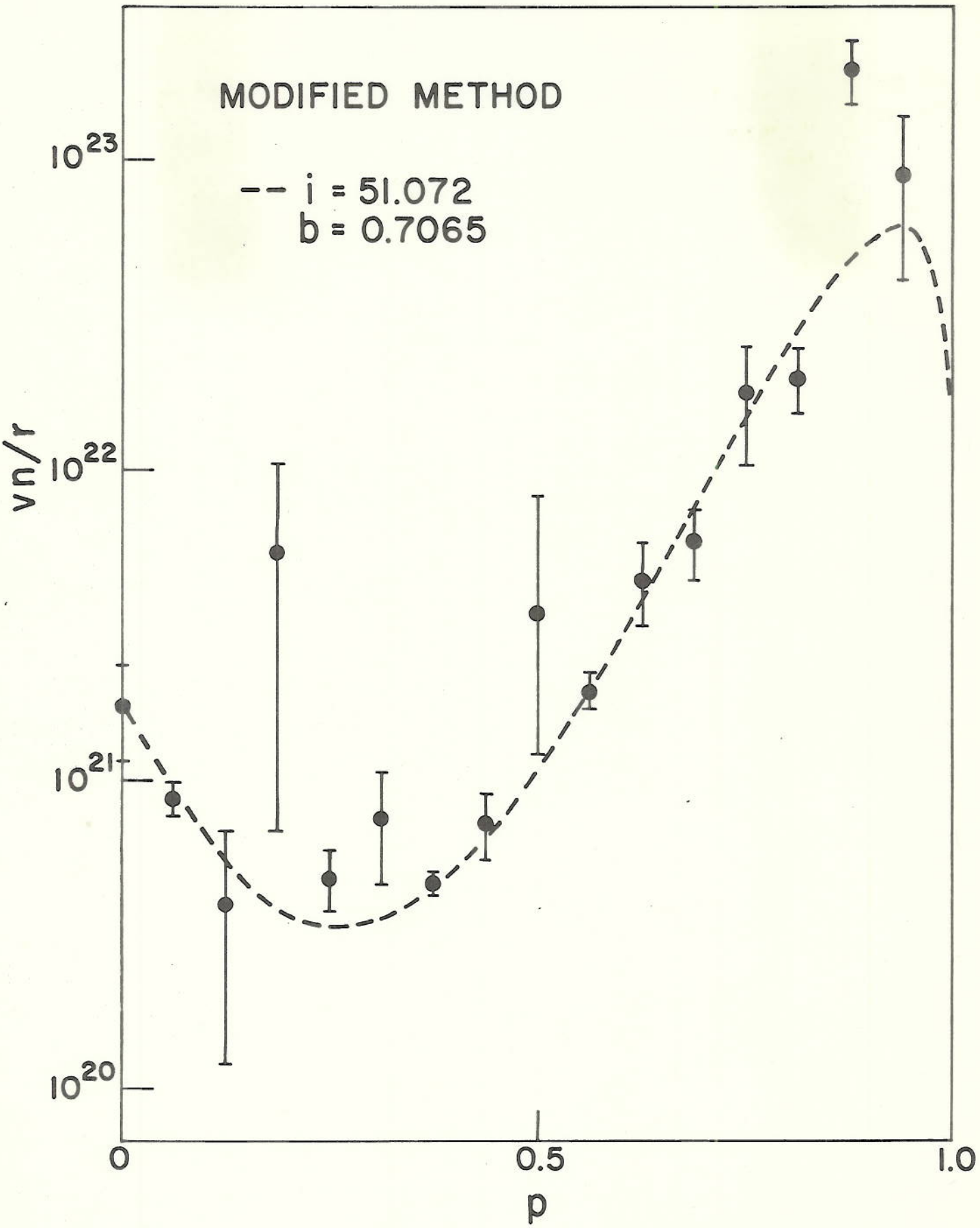
Figure 3. The efficiency in nodes per run times variance of estimate of the basic partial backtrack algorithm as a function of the probability, p, that both branches of the tree are followed. The error bars indicate one standard deviation. A least-squares fit of the theory for T_i trees is shown. Also, for comparison, the dotted curve is for the parameters that best fit the data in figure 4.

Figure 4. The efficiency is p for the modified partial backtrack algorithm along with a least-square fit to the theory for T_i trees.




1
T1
1-b
b
T2
1-b
b(1-b)2
b2(1-b)
b2(1-b)
b3
T3


BASIC METHOD
-- i = 49.105
b = 0.71633
..... i = 51.072
b = 0.7065
10^{24}
10^{23}
10^{22}
10^{21}
vn/r
0
0.5
1.0
p


MODIFIED METHOD
-- i = 51.072
b = 0.7065
10^{23}
10^{22}
10^{21}
10^{20}
vn/r
0
0.5
1.0
p