An Analysis of the Full Alpha-Beta Pruning Algorithm

Gérard M. Baudet

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

An analysis of the alpha-beta pruning algorithm is presented which takes into account both shallow and deep cut-offs. A formula is first developed to measure the average number of terminal nodes examined by the algorithm in a uniform tree of degree n and depth d when ties are allowed among the bottom positions: specifically, all bottom values are assumed to be independent identically distributed random variables drawn from a discrete probability distribution. A worst case analysis over all possible probability distributions is then presented by considering the limiting case when the discrete probability distribution tends to a continuous probability distribution. The branching factor of the alpha-beta pruning algorithm is shown to grow with n as $\Theta(n/\ln n)$, therefore confirming a claim by Knuth and Moore that deep cut-offs only have a second order effect on the behavior of the algorithm.

1 - Introduction

Most so-called intelligent programs use some form of tree searching; among them, most game playing programs are built around an efficient tree searching algorithm known as the *alpha-beta pruning algorithm*. This paper investigates the efficiency of this algorithm with respect to a cost measure first introduced by Knuth and Moore in [3] and given in the following.

Definition 1.1:

Let $N_{n,d}$ be the number of terminal positions examined by some algorithm A in searching a uniform tree of degree n and depth d. The quantity

$$R_A(n) = \lim_{d\to\infty} (N_{n,d})^{1/d}$$

is called the *branching factor* corresponding to the search algorithm A. ■

This research was partly supported by the National Science Foundation under Grant MCS 75-222-55 and the Office of Naval Research under Contract N00014-76-C-0370, NR 044-422 and partly by a Research Grant from the Institut de Recherche d'Informatique et d'Automatique (IRIA), Rocquencourt, France.

Similar analyses have been attempted in two recent papers by Fuller, Gaschnig and Gillogly [1] and by Knuth and Moore [3]. Both papers address the problem of searching a uniform game tree of degree n and depth d with the α-β pruning algorithm under the assumptions that the n^d static values assigned to the terminal nodes are independent identically distributed random variables and that they are *all distinct*. We immediately observe that, in order to evaluate the branching factor, the last assumption requires that the n^d distinct values assigned to the terminal positions be taken from an infinite range. For most practical applications this is, however, unrealistic.

Fuller, Gaschnig and Gillogly developed in [1] a general formula for the average number of terminal positions examined by the α-β procedure. Their formula, however, is computationally intractable and leads to undesirable rounding errors for large trees (i. e., for large n and d) since it involves, in particular, a $2d$-2 nested summation of terms with alternating signs and requires on the order of n^d steps for its evaluation. Then they gave some empirical results based on a series of simulations, and compared the results with actual measurements obtained by running a modified version of the Technology Chess Program [2].

In [3], Knuth and Moore have analyzed, under the same conditions, a simpler version of the full α-β pruning algorithm by not considering the possibility of deep cut-offs; they have shown, in particular, that the branching

factor of the resulting algorithm is $\Theta(n/\ln n)$. Knuth and Moore also considered other assumptions to account for dependencies among the static values assigned to the terminal positions and developed analytic results under those assumptions. Their paper gives, in addition, an excellent presentation and historical account of the α-β pruning algorithm.

Departing from the assumptions of the two papers we just mentioned, we first consider the effect of possible equalities between the values assigned to the terminal nodes of a uniform tree, assuming that these values are independent identically distributed random variables drawn from any *discrete* probability distribution. In Section 2, we establish some notations and preliminary results, and in Section 3, we derive a general formula for the number of terminal nodes examined by the α-β pruning algorithm when we take into account both shallow and deep cut-offs. The evaluation of this formula requires only a finite summation over the range of possible values assigned to the terminal nodes and is relatively easy. We show, in particular, that, when the terminal nodes can only take on two distinct values, the branching factor of the α-β pruning algorithm can grow with n as $O(n/\ln n)$ for some choice of the probability distribution. In Section 4, we show that, when the discrete probability distribution tends to a continuous probability distribution, the summation derived in Section 3 can be replaced by an integral, which constitutes the worst case over all discrete probability distributions. In Section 5, an analysis of this integral shows that the branching factor of the α-β pruning algorithm for a uniform tree of degree n grows with n as $\Theta(n/\ln n)$, therefore confirming a claim by Knuth and Moore [3] that deep cut-offs only have a second order effect on the average behavior of the α-β pruning algorithm. Some concluding remarks and open problems are given in the last section.

2 - Presentation and initial properties of the α-β pruning algorithm

There are two usual approaches for dealing with searching a game tree. In [1], Fuller, Gaschnig and Gillogly adopted the *Min-Max* approach, while, in [3], Knuth and Moore chose the *Nega-Max* approach. We will briefly present, in Section 2.1, the two approaches and introduce the α-β procedure in terms of the Nega-Max model. Then, in Section 2.2, we will reestablish an initial result of [1] which was stated in terms of the Min-Max approach.

2.1 - The α-β procedure

Let us consider a game (like chess, checkers, tic-tac-toe or kalah) played by two players who take turns. It is common to represent the evolution of the game by means of a *game tree*, where each position of the game is represented by a node; if the position is a dead-end, the node is terminal, otherwise all possible moves from that position are represented as the successors of the node. The structure of the tree is preserved by not generating moves leading to some positions already generated (thus, avoiding cycles); this is the function of the *move generator*. The *evaluation function* is another important function in game playing programs; it assigns to each terminal position a *static value* by estimating various parameters such as piece counts, occupation of the board, etc. The evaluation function evaluates the terminal nodes from one player's viewpoint, giving higher values to positions more favorable to this player. It is convenient at this point to name the two players Max and Min. Hence, Max's strategy is to lead the game towards positions with higher values, while Min's strategy is to lead the game towards positions with lower values.

The *minimax procedure* is directly based on this formulation and can be used by either Max or Min to decide on his next move from a given position, assuming that his opponent will respond with his best move. Using a rather

brute force approach, the minimax procedure assigns values to all nodes of a game tree. It first assigns to terminal nodes the results of the evaluation function, then it backs-up to internal nodes corresponding to a position from which it is Max's (Min's) turn to play the maximum (minimum) of the values assigned to its successors.

Suppose it is Max's turn to play from an initial position (corresponding to the root of the game tree), then it is his turn to play from any positions at even depth and Min's turn to play from any positions at odd depth. Therefore, the minimax procedure will back-up values to the nodes of the game tree through a succession of Minimazing/Maximazing operations. This corresponds to the *Min-Max* approach.

By observing that:

$$\max\{ \min\{ x_1, x_2, \dots \}, \min\{ y_1, y_2, \dots \}, \dots \} =$$

$$\max\{ -\max\{ -x_1, -x_2, \dots \}, -\max\{ -y_1, -y_2, \dots \}, \dots \},$$

the Min-Max approach can be directly reformulated into the *Nega-Max* approach. In the Nega-Max formulation, a terminal node of a game tree should be assigned the result of the evaluation function only if it is at an even depth (assuming it is initially Max's turn to play) and it should be assigned the opposite of the result of the evaluation function if it is at an odd depth. The Nega-Max approach requires the same operator at all levels of a game tree, and the uniformity of the notation will make it easier to carry out an analysis. This approach will be used throughout.

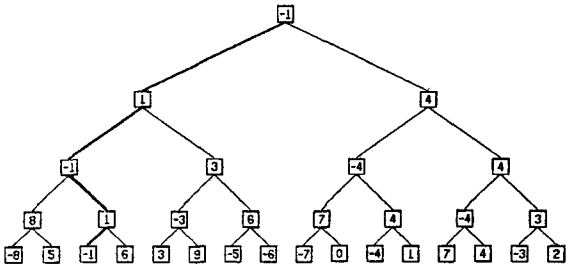

Figure 2.1 - Searching a game tree with the minimax procedure

Figure 2.1 shows the effect of the minimax procedure in a uniform tree of degree *2* and depth *4*. The values assigned to the terminal nodes have been chosen arbitrarily. The path indicated by a darker line shows the sequence of moves selected by the procedure. The minimax procedure is clearly a brute force search and, when exploring a node, it uses none of the information already available from the nodes previously explored. Obviously, by taking advantage of the information previously acquired we can easily improve on the brute force search. Figure 2.2 presents some simple patterns in which the distribution of the information could lead to such improvements.

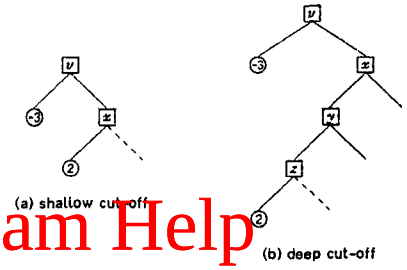

Figure 2.2 - Examples of possible cut-offs

The circled nodes have already been explored, and they are labeled with their backed-up values; the values of the other nodes are yet to be determined. We are interested in the value v of the top level node in both patterns (a) and (b).

Let us consider the pattern of Figure 2.2 (a) first. From the definition of the minimax procedure, the values v and x satisfy:

$$v = \max\{ 3, -x \} , \quad x = \max\{ -2, \dots \} ,$$

which shows that $x \geq -2$ or $2 \geq -x$. Since $3 \geq 2 \geq -x$, it follows that *independent of the exact value of* x, we will have $v = 3$. This shows that we need not explore further the successors of the node labeled x if we are only interested in the value of v. This leads to a first type of *cut-offs* known as *shallow cut-offs*.

The pattern of Figure 2.2 (b) illustrates a *deeper cut-off*. As with the previous example, there are immediate relations between the values of the nodes. In particular, we have

$y \geq -z$, which leads us to consider two cases. Either $y > -z$, and this means that the value y is determined by its right son(s) and certainly does not depend on the right son(s) of z. Or $y = -z$, in which case, since $x \geq -y$ and $z \geq -2$, we deduce $x \geq -2$ or $-x \leq 2$; but since $v = \max\{3, -x\}$ it follows that $v = 3$, independent of the exact value of x and, a fortiori, independent of the exact value of z. This shows that in either case the successors of the node labeled z need not be further explored since the final value of v would in no way be affected.

The two examples presented in Figure 2.2 indicate that a reduction of the search can be achieved if a node passes down to its sons the current value backed-up so far (3 in the case of the two above examples) as a bound for pruning branches 2, 4, 6, ... levels below; the bound can, of course, be improved as the search progresses down the tree (leading to more and more possible cut-offs).

Using two bounds for even and odd levels of a tree, these improvements are implemented in the following procedure adapted from [3].

```
integer procedure
ALPHABETA(position P, integer alpha, integer beta):
    begin integer j, t, n;
    determine the successor positions: P1, ..., Pn;
    if n = 0 then
        ALPHABETA := f(P)
    else
        begin
        for j := 1 step 1 until n do
          begin
          t := -ALPHABETA(Pj,-beta,-alpha);
          if t > alpha then alpha := t;
          if alpha ≥ beta then goto done        (2.1)
          end;
done:   ALPHABETA := alpha
        end
    end
```

The alpha-beta procedure (from [3])

The function denoted by f is the evaluation function which assigns *static values* to terminal positions.

Knuth and Moore [3] have shown this procedure to be correct in the sense that the call ALPHABETA(P,-∞,+∞) assigns to position P the value MINIMAX(P), which is the value assigned by the minimax procedure. More generally, they showed [3, p. 297] that:

ALPHABETA(P,alpha,beta) ≤ alpha,
if MINIMAX(P) ≤ alpha, (2.2)

ALPHABETA(P,alpha,beta) = MINIMAX(P),
if alpha < MINIMAX(P) < beta, (2.3)

ALPHABETA(P,alpha,beta) ≥ beta,
if MINIMAX(P) ≥ beta. (2.4)

The same tree used in Figure 2.1 to illustrate the minimax procedure is shown in Figure 2.3 to illustrate the effects of the α-β procedure.

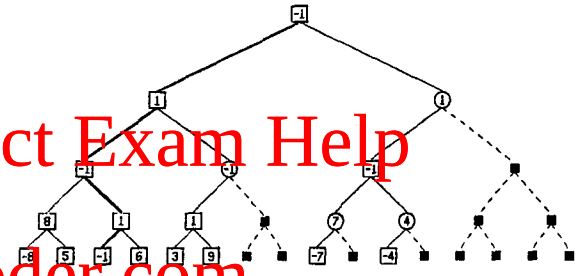

Figure 2.3 - Searching a game tree with the α-β procedure

The branches pruned by the procedure are indicated with dashed lines, and the nodes marked with a circle have not been completely explored. We observe that only 8 out of the 16 terminal positions and 19 out of all the 31 nodes are examined by the α-β pruning algorithm in this example, reducing greatly the cost of searching the tree. As is seen by comparing Figures 2.1 and 2.3, the values backed-up by the α-β procedure to some internal nodes are not necessarily the same as the values backed-up by the minimax procedure, as reflected by the indetermination in equations (2.2) and (2.4). The top value, however, is not affected by this indetermination.

2.2 - Some properties of the α-β pruning algorithm

In this section, we will introduce some notations which will be used throughout, and we will reestablish, in terms of the Nega-Max approach, an initial result of [1] giving a

necessary and sufficient condition for any node of a game tree to be examined by the α-β pruning algorithm.

2.2.1 - Notations

As in [3], we will use the Dewey decimal notation to represent a node in a tree. More precisely, let ϵ, the empty sequence, denote the root of the game tree. Then, if $\mathcal{J}$ denotes some internal node of the tree with n sons, $\mathcal{J}.j$ will denote the j-th son of node $\mathcal{J}$, for $j = 1, ..., n$. In Figure 2.4, node *4.1.3.4.3* is the node at depth 5 whose path from the root is indicated with a darker line.

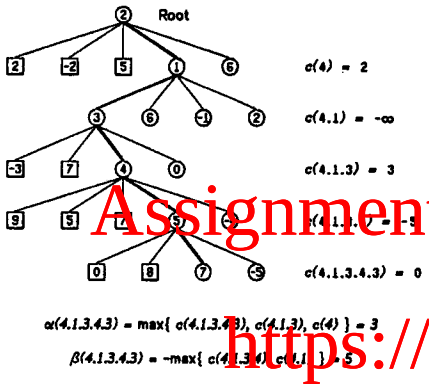

Figure 2.4 - Portion of a game tree showing the path to node <4.1.3.4.3>

The value associated with some node $\mathcal{J}$ of a game tree by the minimax procedure (see Section 2.1) will be denoted by $v(\mathcal{J})$. Then, if $\mathcal{J}$ is a terminal node, $v(\mathcal{J})$ is the *static value* asigned to that terminal position, and, if $\mathcal{J}$ is an internal node, $v(\mathcal{J})$ is the value backed-up to node $\mathcal{J}$ by the minimax procedure. In the latter case, if node $\mathcal{J}$ has n sons, $v(\mathcal{J})$ is given by:

$$v(\mathcal{J}) = \max\{ -v(\mathcal{J}.j) \mid 1 \le j \le n \} . \qquad (2.5)$$

In Figure 2.4, the nodes on the path from the root to node *4.1.3.4.3* are evaluated through formula (2.5) while the other nodes (including *4.1.3.4.3*) are shown as terminal nodes and are assigned arbitrary values. (Nodes are labeled with their values.)

While the values $v(\mathcal{J})$ deal with the static aspect of a game tree, the quantities we will introduce next deal more with the dynamic aspect of the tree when being searched by the α-β procedure.

For any node $\mathcal{J}.j$ at depth $d \ge 1$, we define:

$$c(\mathcal{J}.j) = \max\{ -v(\mathcal{J}.i) \mid 1 \le i \le j-1 \} .$$

(By convention, the maximum over an empty set is defined to be $-\infty$; in particular, $c(\mathcal{J}.1) = -\infty$.) For the root of the tree we also define $c(\epsilon) = -\infty$. The quantity $c(\mathcal{J})$ accounts for the information provided to node $\mathcal{J}$ by its *elder* brothers. These values are indicated to the right of the game tree shown in Figure 2.4 for all nodes on the path to node *4.1.3.4.3*; only the nodes indicated with squares are used in computing these values.

We finally define for any node $\mathcal{J} = j_1. ... .j_d$ at depth $d \ge 1$ in a game tree two quantities directly associated with node $\mathcal{J}$ by the α-β procedure. For $i = 0, ..., d-1$, let $\mathcal{J}_i = j_1. ... .j_{d-i}$. We define:

$$\alpha(\mathcal{J}) = \max\{ c(\mathcal{J}_i) \mid i \text{ is } even, 0 \le i \le d-1 \} ,$$

$$\beta(\mathcal{J}) = -\max\{ c(\mathcal{J}_i) \mid i \text{ is } odd, 0 \le i \le d-1 \} .$$

It is convenient to define these two quantities for the root of the game tree by $\alpha(\epsilon) = -\infty$ and $\beta(\epsilon) = +\infty$ (which is consistent with the definition). These α- and β-values are shown in Figure 2.4 for the node *4.1.3.4.3* along with their definitions.

2.2.2 - Necessary and sufficient condition for a node to be explored by the α-β procedure

The following lemma justifies the notations we just introduced in the preceding section.

Lemma 2.1:

Assume that, initially, the root of a game tree is explored by the α-β procedure through the call

ALPHABETA(root,$-\infty$,$+\infty$) . (2.6)

Then, if node $\mathcal{J}$ is examined, it is through a call of procedure ALPHABETA in which the parameters alpha and beta satisfy:

alpha = $\alpha(\mathcal{J})$, and beta = $\beta(\mathcal{J})$. (2.7)

Proof:

If $\mathcal{J} = j_1 . \ldots . j_d$ denotes some node explored by the procedure at depth $d \geq 1$, let, as before, $\mathcal{J}_i = j_1 . \ldots . j_{d-i}$, for $0 \leq i \leq d-1$. Thus node $\mathcal{J}_1$ is the father of node $\mathcal{J}$, while, if $j_d \geq 2$, node $\mathcal{J}_1.(j_d-1)$ is the brother of $\mathcal{J}$ immediately preceding $\mathcal{J}$ (and explored just before $\mathcal{J}$). Observe that, if $j_d = 1$, $c(\mathcal{J}_0) = c(\mathcal{J}) = -\infty$ and therefore:

$$\begin{aligned} \alpha(\mathcal{J}) &= \max\{\, c(\mathcal{J}_i) \mid i \text{ is } even,\ 0 \leq i \leq d-1 \,\} \\ &= -\,[\, -\max\{\, c(\mathcal{J}_{i+1}) \mid i \text{ is } odd,\ 0 \leq i \leq d-2 \,\}\,] \\ &= -\,\beta(\mathcal{J}_1) \end{aligned}$$

(similarly, $\beta(\mathcal{J}) = -\alpha(\mathcal{J}_1)$). Observe also that, if $j_d \geq 2$:

$$\begin{aligned} \alpha(\mathcal{J}) &= \max\{\, \alpha(\mathcal{J}_2)\ ,\ c(\mathcal{J}) \,\} \\ &= \max\{\, \alpha(\mathcal{J}_2)\ ,\ c[\mathcal{J}_1.(j_d-1)]\ ,\ -v[\mathcal{J}_1.(j_d-1)] \,\} \end{aligned}$$

and that $\beta(\mathcal{J}) = \beta[\mathcal{J}_1.(j_d-1)]$.

By the call of line (2.6), relations (2.7) certainly hold for the root of the game tree, since $\alpha(\epsilon) = -\infty$ and $\beta(\epsilon) = +\infty$. Then the proof follows by induction from inspection of the procedure ALPHABETA, and from the relations we derived above. ∎

The following theorem states a useful relation that characterizes the fact that a node of a tree is explored by the α-β pruning algorithm. This relation was first established by Fuller, Gaschnig and Gillogly [1] with different notations in terms of the Min-Max model.

Theorem 2.1:

Assume that, initially, the root of a game tree is explored by the α-β procedure through the call

ALPHABETA(root,$-\infty$,$+\infty$) .

Then, an arbitrary node $\mathcal{J}$ of the game tree will be subsequently explored if and only if

$$\alpha(\mathcal{J}) < \beta(\mathcal{J}) . \qquad (2.8)$$

Proof:

Because of the presence of line (2.1) in the procedure ALPHABETA, the result follows directly from the result of Lemma 2.1. ∎

Since it will be more convenient in the following sections, rather than $\alpha(\mathcal{J})$ and $\beta(\mathcal{J})$, we will use the quantities:

$$A(\mathcal{J}) = \max\{\, c(\mathcal{J}_i) \mid i \text{ is } even,\ 0 \leq i \leq d-1 \,\} ,$$

$$B(\mathcal{J}) = \max\{\, c(\mathcal{J}_i) \mid i \text{ is } odd,\ 0 \leq i \leq d-1 \,\} ,$$

where $\mathcal{J}_i$ is defined as before. The definitions of $A(\mathcal{J})$ and $B(\mathcal{J})$ are more symmetrical, and relation (2.8) can also be rewritten in a more symmetrical way:

$$A(\mathcal{J}) + B(\mathcal{J}) < 0 . \qquad (2.9)$$

3 - Number of nodes explored by the α-β procedure: discrete case

As in [1] and [3], we will evaluate in this and the following section the amount of work performed in searching a *random uniform game tree* using the α-β pruning algorithm. The definition and some properties of random uniform game trees are given in Section 3.1. The amount of work performed by the α-β procedure is measured by the number of terminal nodes examined during the search and is evaluated in Section 3.2.

3.1 - Random uniform game trees

In order to perform an analysis of the α-β pruning algorithm, we will limit ourselves and consider the following class of game trees.

Definition 3.1:

A game tree in which

(a) all internal nodes have exactly n sons, and

(b) all terminal nodes (or *bottom positions*) are at depth d

is called a *uniform game tree* of degree n and depth d. A uniform game tree which satisfies the additional condition

(c) the values assigned to all terminal nodes (or *bottom values*) are independent identically distributed random variables

is called a *random uniform game tree*, or, for short, a *rug tree*. ∎

Unless otherwise specified, we will only consider throughout a rug tree of degree n and depth d.

Since the value backed-up to a node by the minimax procedure only depends on the backed-up values of its sons, we immediately observe that, by condition (c), the backed-up values of all nodes at the *same* depth are also independent identically distributed random variables. In the remainder of the section, we will assume that the bottom values are drawn from the finite set $\{ k/m \mid -m \le k \le m \}$, for some $m \ge 0$, and we will denote by $\{p_i(k)\}_{-m \le k \le m}$ or simply $\{p_i(k)\}$ the common probability distribution for the backed-up values of all nodes at depth $d - i$ (i. e., $p_i(k)$ is the probability that the value, $v(\mathcal{J})$, backed-up by the minimax procedure to some node $\mathcal{J}$ at depth $d-i$ be k/m). In particular, $\{p_0(k)\}$ is the common probability distribution for all bottom values, and $\{p_d(k)\}$ is the probability distribution for the value backed-up to the root of the rug-tree.

The following lemma states the relations between these probability distributions.

Lemma 3.1:

For $i = 0, ..., d-1$, we have:

$$p_{i+1}(-m) + ... + p_{i+1}(k) = [p_i(-k) + ... + p_i(m)]^n . \qquad (3.1)$$

Proof:

Let $\mathcal{J}$ be some internal node at depth $d-i-1$, then by equation (2.5), $v(\mathcal{J}) \le k$ if and only if $-v(\mathcal{J}.j) \le k$, for $j = 1, ..., n$. Equation (3.1) follows easily from the fact that all variables $v(\mathcal{J}.j)$ are independent. ∎

Since the quantity $p_i(-k) + ... + p_i(m)$ will occur again later on, we define for $i = 0, 1, ...$ and $-m \le k \le m$:

$$\varphi_i(k) = p_i(-k) + ... + p_i(m) .$$

For convenience, we also define $\varphi_i(-m-1) = 0$. Note that $\varphi_i(k)$ is a non-decreasing function of k which satisfies $\varphi_i(-m-1) = 0$ and $\varphi_i(m) = p_i(-m) + ... + p_i(m) = 1$. By rewriting equation (3.1), we see that φ_i satisfies:

$$\varphi_{i+1}(-k-1) = 1 - [\varphi_i(k)]^n \quad \text{for } i = 0, 1, ..., \qquad (3.2)$$

and, therefore:

$$\varphi_{i+2}(k) = 1 - \{1 - [\varphi_i(k)]^n\}^n \quad \text{for } i = 0, 1, ... \qquad (3.3)$$

The following quantities will also be useful in Section 3.2. For $i = 0, 1, ...$ and $-m-1 \le k \le m$, define:

$$\rho_i(k) = 1 + [\varphi_i(k)] + ... + [\varphi_i(k)]^{n-1} , \qquad (3.4)$$

and

$$\sigma_i(k) = 1 + [\varphi_i(-k-1)] + ... + [\varphi_i(-k-1)]^{n-1} . \qquad (3.5)$$

Observe that $\rho_i(-m-1) = \sigma_i(m) = 1$ and $\rho_i(m) = \sigma_i(-m-1) = n$.

Lemma 3.1 establishes the probability distributions for all the values in the nodes of a rug tree. The next lemma establishes a similar result for the quantities $c(\mathcal{J})$ defined in Section 2.

Lemma 3.2:

Let $\mathcal{J}.j$ denote any node at depth i, where $i = 1, ..., d$. If $j = 1$, $c(\mathcal{J}.j) = -\infty$. If $j \ge 2$, then the probability distribution of $c(\mathcal{J}.j)$, denoted by $\{q_k(\mathcal{J}.j)\}_{-m \le k \le m}$, satisfies:

$$q_{-m}(\mathcal{J}.j) + ... + q_k(\mathcal{J}.j) = [\varphi_{d-i}(k)]^{j-1} . \qquad (3.6)$$

Proof:

When $j = 1$, $c(\mathcal{J}.j) = -\infty$ by definition. When $j \ge 2$, equation (3.6) follows from the same argument given in the proof of Lemma 3.1. ∎

In order to evaluate, through equation (2.9), the probability that a terminal node is explored, we first need to determine the probability distributions for the two quantities $A(\mathcal{J})$ and $B(\mathcal{J})$. This is done in the following.

Lemma 3.3:

Let $\mathcal{J} = j_{d-1} \cdot ... \cdot j_1 \cdot j_0$ denote any terminal node.

(1) If $j_i = 1$ for all *even* integers i in the range $0 \le i \le d-1$, then $A(\mathcal{J}) = -\infty$.

(2) Otherwise, the probability distribution for $A(\mathcal{J})$, denoted by $\{a_k(\mathcal{J})\}_{-m \le k \le m}$, satisfies:

$$a_{-m}(\mathcal{J}) + ... + a_k(\mathcal{J}) = \prod_e [\varphi_i(k)]^{j_i - 1} , \qquad (3.7)$$

where the product denoted by $\prod_e$ is extended to all *even* integers in the range $0 \le i \le d-1$.

Similarly,

(1') If $j_i = 1$ for all *odd* integers i in the range $1 \le i \le d-1$, then $B(\mathcal{J}) = -\infty$.

(2') Otherwise, the probability distribution for $B(\mathcal{J})$, denoted by $\{b_k(\mathcal{J})\}_{-m \le k \le m}$, satisfies:

$$b_{-m}(\mathcal{J}) * \ldots * b_k(\mathcal{J}) = \prod_o [\varphi_i(k)]^{j_i - 1}, \qquad (3.8)$$

where the product denoted by $\prod_o$ is extended to all *odd* integers in the range $1 \le i \le d-1$.

Proof:

We will only consider $A(\mathcal{J})$ since the proof relative to $B(\mathcal{J})$ is the same. Part (1) follows directly from the definition. For part (2), let $\mathcal{J}_i$ denote the node $j_{d-1} \cdot \ldots \cdot j_i$. We note that $A(\mathcal{J}) \le k$ if and only if $c(\mathcal{J}_i) \le k$ for all even integers i in the range $0 \le i \le d-1$ such that $j_i \ge 2$. Since the variables $c(\mathcal{J}_i)$ are independent, equation (3.7) follows from equation (3.6) by observing that in the product $\prod_e$ a factor corresponding to $j_i = 1$ amounts to 1. ∎

The last lemma in this section states the probability of exploring a terminal node.

Lemma 3.4:

Let $\mathcal{J} = j_{d-1} \cdot \ldots \cdot j_1 \cdot j_0$ denote any terminal node. The probability $\pi(\mathcal{J})$ that node $\mathcal{J}$ is examined by the α-β procedure is given by $\pi(\mathcal{J}) = 1$ if $j_i = 1$ for all *even* integers i in the range $0 \le i \le d-1$, or for all *odd* integers i in the range $1 \le i \le d-1$, and by

$$\pi(\mathcal{J}) = \sum_{-m \le k \le m-1} a_k(\mathcal{J}) \, [b_{-m}(\mathcal{J}) * \ldots * b_{-k-1}(\mathcal{J})] \qquad (3.9)$$

otherwise.

Proof:

When $j_i = 1$ for all even integers i in the range $0 \le i \le d-1$, by Lemma 3.3 $A(\mathcal{J}) = -\infty$. Hence $A(\mathcal{J}) * B(\mathcal{J}) = -\infty$ too, and by Theorem 2.1 node $\mathcal{J}$ is certainly explored. Similarly when $j_i = 1$ for all odd integers in the range $1 \le i \le d-1$.

Otherwise, both $A(\mathcal{J})$ and $B(\mathcal{J})$ are finite. Let $A(\mathcal{J}) = x_k$. We observe that $A(\mathcal{J}) * B(\mathcal{J}) < 0$ if and only if $-m \le k \le m-1$ and $-x_m \le B(\mathcal{J}) \le x_{-k-1}$. Hence, equation (3.9) follows from Theorem 2.1 and the fact that $A(\mathcal{J})$ and $B(\mathcal{J})$ are independent variables. ∎

Using equations (3.7) and (3.8), equation (3.9) can be rewritten as:

$$\pi(\mathcal{J}) = \sum_{-m \le k \le m-1} a_k(\mathcal{J}) \prod_o [\varphi_i(-k-1)]^{j_i - 1},$$

$$\pi(\mathcal{J}) = \sum_{-m \le k \le m-1} \{\prod_e [\varphi_i(k)]^{j_i - 1} - \prod_e [\varphi_i(k-1)]^{j_i - 1}\} \times \prod_o [\varphi_i(-k-1)]^{j_i - 1} \qquad (3.10)$$

(recall that $\varphi_i(-m-1) = 0$).

3.2 - Number of terminal nodes examined by the α-β pruning algorithm: discrete case

We are now able to evaluate the amount of work performed by the α-β procedure while searching a rug tree. As in [1] and [3], we have chosen to measure the amount of work by the number of terminal nodes examined by the procedure. (We will also consider briefly, at the end of the section, the total number of internal and terminal nodes explored by the procedure as a measure of performance.)

Theorem 3.1:

The average number, $N_{n,d}(m)$, of bottom positions examined by the α-β procedure in searching a rug tree of degree n and depth d, for which the bottom values are distributed according to the discrete probability distribution $\{p_0(k)\}_{-m \le k \le m}$, is given by:

$$N_{n,d}(m) = n^{\lfloor d/2 \rfloor} + \sum_{-m \le k \le m} [\prod_e \rho_i(k) - \prod_e \rho_i(k-1)] \prod_o \sigma_i(k), \qquad (3.11)$$

where the quantities $\rho_i(k)$ and $\sigma_i(k)$ are defined by equations (3.4) and (3.5), and where the products denoted by $\prod_e$ and $\prod_o$ are defined in Lemma 3.3.

Proof:

By definition of the probability $\pi(\mathcal{J})$, the average number of bottom positions examined by the α-β procedure is

$$N_{n,d}(m) = \sum \pi(\mathcal{J}),$$

where the sum is extended to all terminal nodes

$\mathcal{J} = j_{d-1}. \ldots .j_1.j_0$, and is actually a d-nested summation over the range $1 \le j_0 \le n$, $1 \le j_1 \le n, \ldots,$ $1 \le j_{d-1} \le n$. The summation can be rearranged as:

$$N_{n,d}(m) = \sum_e \pi(\mathcal{J}) + \sum_o \pi(\mathcal{J}) + \sum' \pi(\mathcal{J}) - \pi(1. \ldots .1) ,$$

where the three summations $\sum_e$, $\sum_o$ and $\sum'$ correspond to the three expressions for $\pi(\mathcal{J})$ given in Lemma 3.4. The fourth term $\pi(1. \ldots .1)$ is subtracted from the sum since it is counted by both $\sum_e$ and $\sum_o$. These two sums are easily evaluated since all the terms $\pi(\mathcal{J})$ are 1. As $\pi(1. \ldots .1)$ itself is 1, we obtain:

$$N_{n,d}(m) = n^{\lceil d/2 \rceil} + n^{\lfloor d/2 \rfloor} - 1 + \sum' \pi(\mathcal{J}) . \qquad (3.12)$$

It is to be noted that the first three terms correspond exactly to the number of terminal nodes examined by the α-β procedure under optimal ordering of the bottom values (see [5, p. 201]).

We now evaluate the sum $\sum'$. Inside the sum the terms $\pi(\mathcal{J})$ can be evaluated through equation (3.10). We note that all the summations relative to j_i for $i = 0, 1, \ldots, d-1$ can be done independently, each one being the sum of a geometric series. Using the quantities $\rho_i(k)$ and $\sigma_i(k)$ defined by equations (3.4) and (3.5), we obtain:

$$\sum' \pi(\mathcal{J}) = \sum_{-m \le k \le m-1} [\prod_e \rho_i(k) - \prod_e \rho_i(k-1)] \prod_o \sigma_i(k) - \prod_e \rho_i(m-1) + 1 .$$

The theorem follows from this last equation and equation (3.12), using the facts that $\rho_i(m) = n$ and that $\sigma_i(m) = 1$. ∎

The formula of equation (3.11) can be easily evaluated and provides us with a measure of performance for the α-β pruning algorithm. For some applications, however (especially when the cost of generating moves is greater than the cost of evaluating positions), it is more convenient to use the total number of nodes (internal and terminal) explored by the procedure as a measure of performance. Let $T_{n,d}(m)$ denote the average of this number. The same way we evaluated $N_{n,d}(m)$, we can evaluate $T_{n,d}(m)$ by summing the probabilities $\pi(\mathcal{J})$ over all nodes of the tree. We obtain:

$$T_{n,d}(m) = N^0_{n,d}(m) + N^1_{n,d}(m) + \ldots + N^d_{n,d}(m) ,$$

where $N^i_{n,d}(m)$ is the average number of nodes examined at depth i, and is directly derived from the expression of $N_{n,d}(m)$ in equation (3.11) by replacing d by i and $\{p_0(k)\}$ by $\{p_{d-i}(k)\}$ (recall that $\{p_0(k)\}$ is the probability distribution for the values assigned to the terminal nodes and that $\{p_{d-i}(k)\}$ is the probability distribution for the values backed-up to nodes at depth i).

3.3 - Bi-valued rug trees

Although it is relatively easy in most game playing programs to obtain (by inspection of the evaluation function) an accurate bound for the range of distinct values assigned to the various positions of the game, it is usually not so easy to derive a good estimate for the probability distribution of these values. In the remainder of the section we will study rug trees in which the terminal nodes can only take on two distinct values, and we will see, in particular, that a change in the probability distribution of these values can lead to very important differences in the growth rate of $N_{n,d}(m)$.

We will assume in the following that the values assigned to the terminal nodes of a rug tree can only be either -1 or $+1$ with respective probabilities $1-p$ and p, for some $p \in [0, 1]$. Under these conditions, the number, $T_{n,d}(p)$, of terminal nodes examined by the α-β procedure can be obtained as a particular case of equation (3.11) in which $m = 1$ and $\{p_0(k)\}_{-m \le k \le m}$ is defined by $p_0(-1) = 1-p$, $p_0(0) = 0$, $p_0(1) = p$.

Theorem 3.2:

Let $p_0 = p$, and, for $i = 1, 2, \ldots,$ let $p_i = 1 - p^n_{i-1}$.

$$T_{n,d}(p) = n^{\lceil d/2 \rceil} + n^{\lfloor d/2 \rfloor} - 1 + (P_e - 1)(P_o - 1) , \qquad (3.13)$$

with

$$P_e = \prod_e \frac{p_{i+1}}{1 - p_i} , \quad P_o = \prod_o \frac{p_{i+1}}{1 - p_i} ,$$

where the products Π_e and Π_o are defined as before.

Proof:

Choose $m = 1$ and define the probability distribution $\{p_0(k)\}_{-m \le k \le m}$ by $p_0(-1) = 1-p$, $p_0(0) = 0$ and $p_0(1) = p$. Hence $\varphi_0(-2) = 0$, $\varphi_0(-1) = \varphi_0(0) = p = p_0$ and $\varphi_0(1) = 1$. By equation (3.2) we obtain, for $i = 0, 1, \ldots$:

$$\varphi_i(-2) = 0\ ,\quad \varphi_i(-1) = \varphi_i(0) = p_i\ ,\quad \varphi_i(1) = 1\ ,$$

Then equation (3.13) follows directly from Theorem 3.1 and equations (3.4) and (3.5). ∎

Equation (3.13) can be evaluated very easily and, in particular, we note that for $0 < p < 1$:

$$T_{n,d}(p) > T_{n,d}(0) = T_{n,d}(1) = n^{\lfloor d/2 \rfloor} + n^{\lfloor d/2 \rfloor} - 1\ . \qquad (3.14)$$

This last equation shows that $T_{n,d}(p)$ reaches its minimum $n^{\lfloor d/2 \rfloor} + n^{\lfloor d/2 \rfloor} - 1$ for $p = 0$ and $p = 1$. This is in agreement with the result of Slagle and Dixon [5, p. 201] since it corresponds to the case when all terminal nodes are assigned the same value and therefore all possible cut-offs do occur. Equation (3.14) also shows that $T_{n,d}(p)$ admits a maximum for $p \in (0, 1)$; although the exact maximum cannot be readily obtained, we will derive a lower bound in the following. We first establish a preliminary result.

Lemma 3.5:

The unique positive root, ξ_n, of the equation

$$x^n + x - 1 = 0$$

is in the interval $(0, 1)$. Asymptotically (for large n) it satisfies:

$$1 - \xi_n \sim \frac{1}{n} \ln n\ . \qquad (3.15)$$

Proof:

As there is no ambiguity, we will drop the index n from ξ_n in the following.

Let $g(x) = x^n + x - 1$, note that $g(0) = -1 < 0$ and $g(1) = 1 > 0$. Since $g(x)$ is continuous and strictly increases for x positive, the equation $g(x) = 0$ admits a unique positive root, ξ, which is in the interval $(0, 1)$.

We observe that equation $\xi^n + \xi - 1 = 0$ can be rewritten as

$$1 - \xi = \frac{1}{1 + (1 + \xi + \ldots + \xi^{n-1})}\ ,$$

from which we deduce that

$$1 - \xi > \frac{1}{n+1}\ . \qquad (3.16)$$

On the other hand, since $\xi^n = 1 - \xi$, we obtain

$$n(\xi - 1) > n \ln \xi = \ln(1-\xi),$$

which shows, along with equation (3.16), that

$$1 - \xi < \frac{1}{n} \ln(n+1) = \frac{1}{n} \ln n + O(n^{-2})\ . \qquad (3.17)$$

Similarly, taking the logarithm of both sides of equation (3.17), and using the facts that $1 - \xi = \xi^n$ and that $\ln \xi > 1 - \frac{1}{\xi}$, we obtain:

$$\xi < \frac{1}{1 + \ln(n/\ln n + 1)}\ ,$$

hence:

$$1 - \xi > \frac{1}{n} \ln(n/\ln n + 1) + O[(\frac{1}{n} \ln n)^2] = \frac{1}{n} \ln n + O(\frac{1}{n} \ln \ln n)\ .$$

Equation (3.15) follows directly from the previous equation and equation (3.17). ∎

When $p = \xi_n$ we obtain immediately that, for $i = 0, 1, \ldots$, $p_i = \xi_n$. Hence

$$P_e = [\xi_n/(1-\xi_n)]^{\lfloor d/2 \rfloor} \quad \text{and} \quad P_o = [\xi_n/(1-\xi_n)]^{\lfloor d/2 \rfloor}\ .$$

From equations (3.13) and (3.15) it follows that, for large n:

$$T_{n,d}(\xi_n) \sim [n/\ln n]^d\ , \qquad (3.18)$$

while equation (3.14) shows that

$$T_{n,d}(0) = T_{n,d}(1) \sim O(n^{\lfloor d/2 \rfloor})\ . \qquad (3.19)$$

Equations (3.18) and (3.19) indicate that $T_{n,d}(p)$ can be largely influenced by the variations of the probability distribution for the static values. This result can be easily generalized to $N_{n,d}(m)$. In the next section, we will derive an approximation to $N_{n,d}(m)$ which corresponds to its worst case behavior.

4 - Number of nodes explored by the α-β procedure: continuous case

In this section, we derive an approximation to $N_{n,d}(m)$ by considering the limit of the finite series of

equation (3.11) when m tends to infinity while the discrete probability distribution $\{p_0(k)\}_{-m\le k\le m}$ tends to a continuous probability distribution. This corresponds to the case studied by Fuller, Gaschnig and Gillogly [1] and by Knuth and Moore [3] when the terminal nodes of a rug tree are all assigned distinct values. In particular, we will reestablish (with a much simpler formula) a result of [1].

4.1 - Notations and preliminary results

We first introduce the sequence of functions $\{f_i\}$ mapping the interval $[0, 1]$ into itself, and defined recursively by:

$$f_0(x) = x ,$$

$$f_i(x) = 1 - \{1 - [f_{i-1}(x)]^n\}^n \quad \text{for } i = 1, 2, \ldots .$$

It is readily verified by induction on i that all functions f_i are strictly increasing on $[0, 1]$ and satisfy $f_i(0) = 0$ and $f_i(1) = 1$, i. e., 0 and 1 are two fixed points of the functions f_i, for all n and i. The function f_i will be shown to be related to the quantities $\varphi_{2i}(k)$ defined in Section 3.1. Similarly, in relation to the quantities $\varphi_{2i}(k)$ and $\varphi_{2i+1}(k)$, we define the following functions on $[0, 1]$: for $i = 1, 2, \ldots$, let

$$r_i(x) = \frac{1-[f_{i-1}(x)]^n}{1-f_{i-1}(x)} ,$$

$$s_i(x) = \frac{f_i(x)}{[f_{i-1}(x)]^n} .$$

If we define $r_i(1) = n$ and $s_i(0) = 1$, we observe that all functions r_i and s_i are continuous on $[0, 1]$ (they are actually polynomials in x), and that r_i is strictly increasing while s_i is strictly decreasing.

In relation to the two products $\prod_e$ and $\prod_o$, we also introduce, for $i = 1, 2, \ldots$, the following functions on $[0, 1]$:

$$R_i(x) = r_1(x) \times \ldots \times r_{\lfloor i/2 \rfloor}(x) ,$$

$$S_i(x) = s_1(x) \times \ldots \times s_{\lfloor i/2 \rfloor}(x) ,$$

where $S_1(x) = 1$. Observe here, too, that functions R_i and S_i are polynomials, and that, when x increases from 0 to 1, $R_i(x)$ increases from 1 to $n^{\lfloor i/2 \rfloor}$ while $S_i(x)$ decreases from $n^{\lfloor i/2 \rfloor}$ to 1.

Lastly, for $k = 0, 1, \ldots, 2m+1$, let

$$\tau_k = \varphi_0(k-m-1) .$$

Lemma 4.1:

For $i = 1, 2, \ldots$ and $k = 0, \ldots, 2m+1$, we have:

$$r_i(\tau_k) = \rho_{2i-2}(k-m-1) , \tag{4.1}$$

$$s_i(\tau_k) = \sigma_{2i-1}(k-m-1) . \tag{4.2}$$

Proof:

The proof follows immediately from the definitions by induction on i. ∎

4.2 - Number of bottom positions examined by the α-β procedure: continuous case

Let us return to the definition of the sequence $T_m = \{\tau_k\}_{0\le k\le 2m+1}$. As was observed in Section 3.1 with the sequence $\{\varphi_i(k)\}$, the sequence T_m is non-decreasing and defines a partition of the interval $[0, 1]$, i. e.:

$$0 = \tau_0 \le \tau_1 \le \ldots \le \tau_{2m} \le \tau_{2m+1} = 1 .$$

The norm of the partition T_m is

$$\|T_m\| = \max\{ \tau_k - \tau_{k-1} \mid 1 \le k \le 2m+1 \}$$

$$= \max\{ p_0(k) \mid -m \le k \le m \} .$$

In the remainder of the section we require the following.

Assumption:

(A1) $\lim_{m\to\infty} \max\{ p_0(k) \mid -m \le k \le m \} = 0.$ ∎

This assumption ensures that the norm of the partition T_m tends to 0 when m tends to infinity. It also shows that, as m tends to infinity, the probability of two terminal nodes being assigned the same value vanishes. This corresponds to the case studied by Fuller, Gaschnig and Gillogly [1], and by Knuth and Moore [3].

With this assumption, we will now see that the finite series of equation (3.11) can be replaced by an integral when $m\to\infty$. This is established in the following.

Theorem 4.1:

Under assumption (A1), we have:

$$\lim_{m\to\infty} N_{n,d}(m) = n^{\lfloor d/2 \rfloor} + \int_0^1 R_d(t).S_d(t).dt , \tag{4.3}$$

where $R'_d(x)$ is the first derivative of $R_d(x)$.

Proof:

Since there is no risks of confusion, we will drop, in the following, the index d from the functions R_d and S_d.

It follows directly from Lemma 4.1 that, for $k = 0, ..., 2m+1$:

$$R(\tau_k) = \prod_e \rho_i(k-m-1) ,$$

$$S(\tau_k) = \prod_o \sigma_i(k-m-1) ,$$

which shows that equation (3.11) can be simply rewritten as:

$$N_{n,d}(m) = n^{\lfloor d/2 \rfloor} + \sum_{1 \le k \le 2m+1} [R(\tau_k) - R(\tau_{k-1})] S(\tau_k) .$$

Let A_m denote the series defined in this last equation.

Recall that $R(x)$ is a polynomial. By considering the Taylor development of $R(\tau_{k-1})$, we obtain for $k = 1, ..., 2m+1$:

$$R(\tau_k) - R(\tau_{k-1}) = [\tau_k - \tau_{k-1}] R'(\tau_k) + \frac{1}{2} [\tau_k - \tau_{k-1}]^2 R''(t_k) ,$$

where $\tau_{k-1} \le t_k \le \tau_k$. Hence:

$$A_m = \sum_{1 \le k \le 2m+1} [\tau_k - \tau_{k-1}] R'(\tau_k) S(\tau_k) + \sum_{1 \le k \le 2m+1} \frac{1}{2} [\tau_k - \tau_{k-1}]^2 R''(t_k) S(\tau_k) \qquad (4.4)$$

Since R and S are polynomials, the quantity $|R''(x)S(y)/2|$ is bounded by some constant, say M, for any x and y in $[0, 1]$. In particular, the second sum in equation (4.4) is bounded in module by $M.\|T_m\|.[\tau_{2m+1} - \tau_0] = M.\|T_m\|$ and therefore tends to 0 when $m \to \infty$ since, from assumption (A1), $\|T_m\| \to 0$.

As for the first sum in equation (4.4), we observe that it corresponds to a Riemann sum for the function $R'(x)S(x)$ over the partition T_m of $[0, 1]$. Therefore since, in particular, this function is continuous and since $\|T_m\|$ tends to 0, the sum tends to the integral of equation (4.3). This proves the theorem. ∎

In the remainder of the section we will reinterpret the limit of $N_{n,d}(m)$ established in Theorem 4.1.

Let G be the distribution function of some continuous probability density function g, and assume, to simplify the discussion, that $G(-1) = 0$ and $G(1) = 1$ (therefore, $G(x) = 0$ for $x \le -1$ and $G(x) = 1$ for $x \ge 1$). We define a sequence of functions G_m for $m = 0, 1, ...$ as follows. For $-m \le k \le m$, let $x_k = k/m$. Function G_m is defined as the following step function:

$$G_m(x) = \begin{cases} 0 & \text{if } x < x_{-m} = 0, \\ G(x_k) & \text{if } x_k \le x < x_{k+1}, \text{ for } -m \le k \le m-1, \\ 1 & \text{if } 1 = x_m \le x. \end{cases}$$

The sequence of functions $\{G_m\}$ constitutes a sequence of approximations to the continuous function G. (It should be noted that the convergence of the sequence is uniform on the interval $[0, 1]$.) The function G_m corresponds to the cumulative distribution of the discrete probability distribution $p_0(k) = G_m(x_k^+) - G_m(x_k^-)$ associated with the points $x_k = k/m$, for $k = -m, ..., m$.

Using the approximation $\{p_0(k)\}_{-m \le k \le m}$ to the density function g, equation (3.11) provides us with an approximation to the average number of bottom positions examined by the α-β procedure in a rug tree in which the bottom values are drawn from the continuous probability density function g. When m becomes larger, the approximation becomes better, and (due to the uniform convergence of the sequence G_m) it can actually be shown (in a rather technical way) that the limit of $N_{n,d}(m)$ when $m \to \infty$ corresponds exactly to the average number of bottom positions examined by the α-β procedure in the continuous case. As a matter of fact, equation (4.3) could be derived directly by considering a continuous probability distribution rather than a discrete one in very much the same way we derived equation (3.11) in Section 3. This result is stated in the following.

Theorem 4.2:

Let $f_0(x) = x$, and, for $i = 1, 2, ...$, define:

$$f_i(x) = 1 - \{1 - [f_{i-1}(x)]^n\}^n ,$$

$$r_i(x) = \frac{1-[f_{i-1}(x)]^n}{1-f_{i-1}(x)} ,$$

$$s_i(x) = \frac{f_i(x)}{[f_{i-1}(x)]^n} ,$$

$$R_i(x) = r_1(x) \times \ldots \times r_{\lfloor i/2 \rfloor}(x) ,$$

$$S_i(x) = s_1(x) \times \ldots \times s_{\lfloor i/2 \rfloor}(x) .$$

The average number, $N_{n,d}$, of terminal nodes examined by the α-β pruning algorithm in a rug tree of degree n and depth d for which the bottom values are drawn from a continuous distribution is given by:

$$N_{n,d} = n^{\lfloor d/2 \rfloor} + \int_0^1 R'_d(t).S_d(t).dt . \qquad (4.5)$$

It is to be noted that, unlike the case of a discrete probability distribution, when the bottom values are drawn from a continuous distribution, the number of terminal positions examined by the α-β procedure does not depend on the distribution function.

4.3 - Discrete case versus continuous case

Since equation (4.5) has been derived as the limit of equation (3.11), it is reasonable to investigate the validity of the approximation of $N_{n,d}(m)$ by $N_{n,d}$. As was seen in Section 3.3, $N_{n,d}(m)$ strongly depends on the probability distribution $\{p_0(k)\}_{-m \le k \le m}$ and, therefore, we cannot expect $N_{n,d}$ to be a close approximation of $N_{n,d}(m)$ in all cases. We will see below, however, that $N_{n,d}$ provides us with a good insight into the behavior of the α-β pruning algorithm. Namely, we will see that it constitutes the worst case of $N_{n,d}(m)$ over all discrete probability distributions.

Since $N_{n,d}$ was obtained as the limit of $N_{n,d}(m)$, it is sufficient to show that, for all probability distributions $\{p_0(k)\}_{-m \le k \le m}$, we have:

$$N_{n,d} \ge N_{n,d}(m) . \qquad (4.6)$$

In order to prove inequality (4.6), it is convenient to give a geometric interpretation of both $N_{n,d}$ and $N_{n,d}(m)$.

Consider the curve $(\mathcal{L})$ defined by the Cartesian coordinates (x, y) through the parametric equations

$$(\mathcal{L}): \quad [\, x = R_d(t) , \; y = S_d(t) \,] ,$$

where the parameter t varies in the interval $[0, 1]$. The integral of equation (4.5) represents the area delimited by the curve $(\mathcal{L})$, the x-axis and the parallels to the y-axis at the abscissas $R_d(0) = 1$ and $R_d(1) = n^{\lfloor d/2 \rfloor}$ (see Figure 4.1). Since $R_d(0) = 1$ and $S_d(0) = n^{\lfloor d/2 \rfloor}$, the term $n^{\lfloor d/2 \rfloor}$ of equation (4.5) can be accounted for by the area of the rectangle delimited by the x-axis, the y-axis and the lines $x = 1$ and $y = n^{\lfloor d/2 \rfloor}$ (the latter line extends the curve $(\mathcal{L})$ in a continuous way). Figure 4.1 represents the curve $(\mathcal{L})$ and its *extension* in the case $n = 3$, $d = 6$. The area below the unbroken lines represents the quantity $N_{n,d}$.

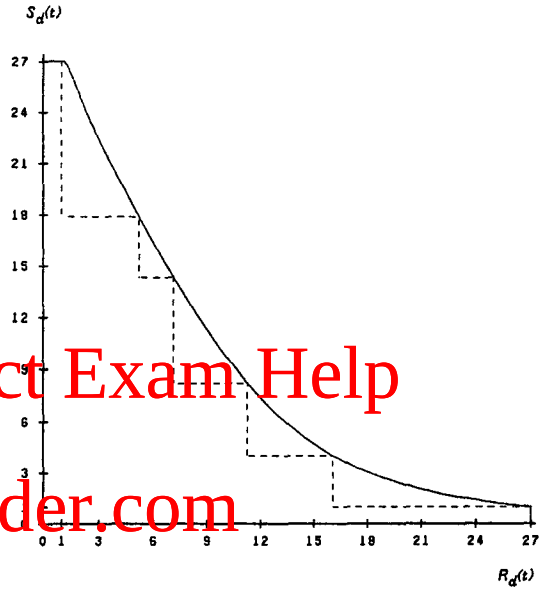

Figure 4.1 - Geometric interpretation of $N_{n,d}$ and $N_{n,d}(m)$

The sum of equation (3.11) can also be represented along with the curve $(\mathcal{L})$. It follows directly from the relations of equations (4.1) and (4.2) that the terms of the sum represent the areas of the rectangles delimited by the lines $x = R(\tau_{k-1})$, $x = R(\tau_k)$, $y = 0$ and $y = R(\tau_k)$, for $k = 1, 2, \ldots, 2m-1$. The quantity $N_{n,d}(m)$ represents therefore the area of Figure 4.1 shown below the broken lines.

Inequality (4.6), then, follows directly from the fact that, when t increases in $[0, 1]$, $R(t)$ increases while $S(t)$ decreases.

5 - On the branching factor of the α-β pruning algorithm

We have deliberately chosen to introduce first the case

when the bottom values of a game tree are drawn from a discrete probability distribution since it is of most interest in actual applications. The case of a continuous distribution, however, lends itself to an easier analysis, and, since it constitutes the worst case over all discrete probability distributions, we will, in this section, examine the integral of equation (4.5) rather than the series of equation (3.11).

5.1 - Previous results

In Section 1, we introduced the branching factor as a cost measure for the work involved in searching a tree. Rather than considering the number, $N_{n,d}$, of terminal positions examined by a search algorithm, as a measure of performance of the algorithm, we could have considered the total number, $T_{n,d}$, of nodes (terminal and internal) explored during the search. In the case of the α-β pruning algorithm, since $N_{n,d}$, given by equation (4.5), does not depend on the distribution function of the bottom values, we deduce that $T_{n,d}$ satisfies:

$$T_{n,d} = 1 + N_{n,1} + \ldots + N_{n,d}$$

Since it can be checked easily that $0 \le N_{n,i-1} \le N_{n,i}$, we obtain $N_{n,d} \le T_{n,d} \le dN_{n,d}$, and therefore:

$$\lim_{d\to\infty} (T_{n,d})^{1/d} = \lim_{d\to\infty} (N_{n,d})^{1/d} = \mathcal{R}_{\alpha\text{-}\beta}(n).$$

Thus Definition 1.1 provides us with a measure of performance useful to compare search algorithms. In the following, we review some of the results which have already been presented in the literature.

Minimax search

The minimax search examines systematically all nodes of a tree. It, therefore, examines $N_{n,d} = n^d$ terminal nodes in a uniform tree of degree n and depth d, leading to a branching factor

$$\mathcal{R}_{minimax}(n) = n .$$

α-β procedure under optimal ordering

Slagle and Dixon [5, p. 201] have shown that, when all possible α- and β-cut-offs occur, the α-β procedure examines

$$N_{n,d} = n^{\lceil d/2 \rceil} + n^{\lfloor d/2 \rfloor} - 1$$

terminal positions. In this case, the corresponding branching factor is

$$\mathcal{R}_{opt}(n) = n^{1/2} .$$

α-β procedure (experimental results from [1])

Based on a series of simulation results, Fuller, Gaschnig and Gillogly [1] have argued that the formula

$$N_{n,d} = c(d).n^{0.72d + 0.277}$$

constitutes a reasonable approximation to the number of bottom positions examined by the α-β procedure for small values of n and d, and that $1 \le c(d) \le 2$ (at least for the range of values they considered). For comparison purposes, let us assume that their approximation can be extrapolated for any n and d. Provided that $c(d)^{1/d} \to 1$ when $d \to \infty$, we obtain

$$\mathcal{R}_{\alpha\text{-}\beta}(n) \sim n^{0.72} .$$

In view of the results of Section 3.3, we can question the accuracy of the approximation for large n since it follows from Theorem 3.2 that

$$\lim_{d\to\infty} [T_{n,d}(\xi_n)]^{1/d} = O(n/\ln n) .$$

α-β procedure without deep cut-offs

Knuth and Moore [3] have analyzed a simpler version of the α-β procedure by not considering the possibilities of deep cut-offs. This *β-procedure* is the same as the α-β procedure except that no α-values are passed to the α-β procedure; instead, the lower value α is always set to $-\infty$ before exploring the successors of a node. Knuth and Moore have shown that the branching factor of this procedure satisfies

$$\mathcal{R}_{\beta}(n) = \Theta(n/\ln n) .$$

Note that, since the β-procedure always explores more nodes at any depth in a tree than the full α-β procedure does in the same tree, $\mathcal{R}_{\beta}(n)$ provides us with an upper bound for $\mathcal{R}_{\alpha\text{-}\beta}(n)$.

5.2 - Bounds on the branching factor of the α-β procedure

In this section we will derive some lower and upper bounds on the branching factor of the α-β pruning algorithm. In particular, since the lower bound we derive grows with n as $n/\ln n$, we will be able to conclude, using the result on the branching factor of the α-β procedure *without* deep cut-offs established by Knuth and Moore in [3], that the branching factor of the α-β procedure is $\Theta(n/\ln n)$.

We introduced in Section 4.1 the sequence of functions f_i, $i = 0, 1, \ldots$, from $[0, 1]$ to itself, and we observed that all functions f_i share the two fixed points 0 and 1 (independent of n). Another common fixed point, which depends on n, was introduced in Section 3.3.

Lemma 5.1:

For a given n, all functions f_i, for $i = 0, 1, \ldots$, share the common fixed point $\xi_n \in (0, 1)$, the unique positive root of the equation

$$x^n + x - 1 = 0 .$$

Proof:

Since $f_0(x) = x$, ξ is certainly a fixed point of f_0; then, the proof follows immediately by induction on i. ∎

Since ξ_n is a fixed point common to all functions f_i, $i = 0, 1, \ldots$, it is easy to evaluate at this point the functions r_i and s_i defined in Section 4.1. For $i = 1, 2, \ldots$, we deduce that:

$$r_i(\xi_n) = s_i(\xi_n) = \xi_n/(1 - \xi_n) . \qquad (5.1)$$

In particular, it follows from Lemma 3.5 that, for large n:

$$r_i(\xi_n) = s_i(\xi_n) \sim n/\ln n . \qquad (5.2)$$

Equations (5.1) and (5.2) will be useful to obtain the desired bounds in the remainder of the section.

The geometric representation of equation (4.5), given in Figure 4.1, makes it easy to derive bounds on the quantity $N_{n,d}$. They are stated in the following.

Theorem 5.1:

The branching factor of the α-β pruning algorithm in the search of a rug tree of degree n satisfies:

$$n/\ln n \sim \xi_n/(1-\xi_n) < \mathcal{R}_{\alpha\text{-}\beta}(n) < \sqrt{n\xi_n/(1-\xi_n)} \sim n/\sqrt{\ln n} , \qquad (5.3)$$

for $n = 2, 3, \ldots$.

Proof:

Since, when t increases in $[0, 1]$, $R_d(t)$ increases while $S_d(t)$ decreases, it follows directly that for any α in $[0, 1]$ we have the following inequalities:

$$R_d(\alpha).S_d(\alpha) < N_{n,d} < R_d(\alpha).S_d(0) + [R_d(1) - R_d(\alpha)].S_d(\alpha) . \qquad (5.4)$$

If we choose $\alpha = \xi_n$, we have $R_d(\alpha) = [\xi_n/(1-\xi_n)]^{\lfloor d/2 \rfloor}$ and $S_d(\alpha) = [\xi_n/(1-\xi_n)]^{\lfloor d/2 \rfloor}$. Since $R_d(1) = n^{\lfloor d/2 \rfloor}$ and $S_d(0) = n^{\lfloor d/2 \rfloor}$, inequality (5.3) follows immediately from inequality (5.4) and the results of Lemma 3.5. ∎

As an immediate consequence, we obtain the following.

Theorem 5.2:

The branching factor of the α-β pruning algorithm in the search of a rug tree of degree n satisfies, for large n:

$$\mathcal{R}_{\alpha\text{-}\beta}(n) = \Theta(n/\ln n) .$$

Proof:

The result comes directly from the lower bound $n/\ln n$ of Theorem 5.1, and from the upper bound $\mathcal{R}_{\beta}(n)$ obtained for the α-β procedure *without* deep cut-offs, which Knuth and Moore have shown to be $\Theta(n/\ln n)$. ∎

This results confirms, as was suggested by Knuth and Moore [3, p. 310], that deep cut-offs have only a second order effect on the behavior of the α-β pruning algorithm. On the other hand, it shows that the formula proposed by Fuller, Gaschnig and Gillogly in [1] and mentioned in Section 5.1, if it constitutes a reasonable approximation for small values of n and d (the range of values they considered is $n + d \leq 12$), is certainly not adequate for large values.

We note that the bounds of Theorem 5.1 were obtained without difficulty by conveniently choosing just one point, ξ_n, on the curve $(\mathcal{L})$ since it was easy to evaluate both $R_d(\xi_n)$ and $S_d(\xi_n)$. In the next section, using a different approach, we will derive a tighter upper bound for $N_{n,d}$, and hence for $\mathcal{R}_{\alpha\text{-}\beta}(n)$.

5.3 - Improved upper bound

Since, for $d = 1, 2, \ldots$, $N_{n,d} \leq N_{n,d+1} \leq nN_{n,d}$, then, if $(N_{n,d})^{1/d}$ tends to some limit when d tends to infinity as an *even* integer, this quantity tends to the same limit when d tends to infinity as an *odd* integer. Therefore, without loss of generality, we will only consider, in this section, the case when d is an even integer. Let $d = 2h$.

For x in $[0, 1]$ and for $i = 1, 2, \ldots$, we define $p_i(x) = r_i(x)s_i(x)$.

Lemma 5.2:

All functions p_i, for $i = 1, 2, \ldots$, have the same absolute maximum, M_n, in the interval $[0, 1]$.

Proof:

From the definitions of $r_i(x)$ and $s_i(x)$ we have, for $i = 1, 2, \ldots$:

$$r_i(x) = r_1[f_{i-1}(x)],$$

and

$$s_i(x) = s_1[f_{i-1}(x)].$$

Therefore, for $i = 1, 2, \ldots$, we also have, from the definition of $p_i(x)$:

$$p_i(x) = p_1[f_{i-1}(x)].$$

The lemma follows by observing that, for $i = 1, 2, \ldots$, f_{i-1} is a one-to-one function from $[0, 1]$ to itself. ∎

Lemma 5.2 shows that, in order to study the maximum of $p_i(x)$, when $x \in [0, 1]$, it is sufficient to study the maximum of the polynomial

$$p_i(x) = \frac{1 - x^n}{1 - x} \frac{1 - (1 - x^n)^n}{x^n}, \quad \text{for } x \in [0, 1].$$

Observe that $M_n \geq p_1(\xi_n) = [\xi_n/(1-\xi_n)]^2$, in particular, since it can be checked easily that, for $n = 2, 3, \ldots$, $\xi_n > \sqrt{n}/(1+\sqrt{n})$, it follows that

$$M_n > n \quad \text{for } n = 2, 3, \ldots . \tag{5.5}$$

Theorem 5.3

The branching factor of the α-β pruning algorithm for a rug tree of degree n satisfies:

$$\mathcal{R}_{\alpha\text{-}\beta}(n) \leq \sqrt{M_n}, \tag{5.6}$$

where M_n is defined in Lemma 5.2.

Proof:

From the definition of $R_{2h}(t)$, we obtain for $h = 2, 3, \ldots$:

$$R'_{2h}(t) = R'_{2h-2}(t).r_h(t) + R_{2h-2}(t).r'_h(t).$$

By multiplication by $S_{2h}(t)$ it follows that

$$R'_{2h}(t).S_{2h}(t) = R'_{2h-2}(t).S_{2h-2}(t).p_h(t) + R_{2h-2}(t).S_{2h-2}(t).r'_h(t).s_h(t).$$

Since, for $t \in [0, 1]$, all factors in this equation are non-negative, we deduce, using the results of Lemma 5.2 and the fact that $s_h(t) \leq n$ when $t \in [0, 1]$, that:

$$R'_{2h}(t)S_{2h}(t) \leq M_n R'_{2h-2}(t)S_{2h-2}(t) + n M_n^{h-1} r'_h(t).$$

Since, in addition,

$$R'_2(t) S_2(t) = r'_1(t) s_1(t) \leq n r'_1(t),$$

it follows that for $t \in [0, 1]$ and $h = 1, 2, \ldots$:

$$R'_{2h}(t)S_{2h}(t) \leq n M_n^{h-1} [r'_1(t) + \ldots + r'_h(t)]. \tag{5.7}$$

Let $I_{n,d}$ be the integral defined in equation (4.5). By integrating inequality (5.7) over $[0, 1]$ we see that $I_{n,d}$ satisfies:

$$I_{n,2h} \leq n M_n^{h-1} [h(n-1)] = n(n-1) h M_n^{h-1}$$

since $r_i(0) = 1$ and $r_i(1) = n$ for $i = 1, 2, \ldots$. This shows that

$$N_{n,2h} \leq n^h + n(n-1) h M_n^{h-1}.$$

Equation (5.6) now follows directly from inequality (5.5). ∎

5.4 - Numerical results

Table 5.1 summarizes the results of this section. It presents the various lower and upper bounds we have derived for the branching factor of the α-β pruning algorithm from equations (5.3) and (5.6). Although we have not been able to give an estimate for the asymptotic growth of $\sqrt{M_n}$, we can easily derive an upper bound for this

quantity by studying rug trees of depth 2 since:

$$M_n \le N_{n,2} \le 2n\xi_n/(1-\xi_n) - [\xi_n/(1-\xi_n)]^2 \sim 2n^2/\ln n ,$$

which shows that $\sqrt{M_n} \le O(n/\sqrt{\ln n})$. The numerical results of Table 5.1 indicate that $\sqrt{M_n}$ is a much better upper bound for $\mathcal{R}_{\alpha\text{-}\beta}(n)$ than $\sqrt{n\xi_n/(1-\xi_n)}$ for the range of values we have considered.

	lower bnd.	upper bounds		
n	$\xi_n/(1-\xi_n)$	$\sqrt{M_n}$	$\sqrt{n\xi_n/(1-\xi_n)}$	from [3]
2	1.618	1.622	1.799	1.884
3	2.148	2.168	2.538	2.666
4	2.630	2.678	3.243	3.397
5	3.080	3.166	3.924	4.095
6	3.506	3.638	4.587	4.767
7	3.915	4.098	5.235	5.421
8	4.309	4.549	5.872	6.059
9	4.692	4.993	6.498	6.684
10	5.064	5.430	7.116	7.298
11	5.427	5.862	7.726	7.902
12	5.782	6.290	8.330	8.498
13	6.130	6.713	8.927	9.086
14	6.473	7.133	9.519	9.668
15	6.809	7.549	10.107	10.243
16	7.141	7.963	10.689	10.813
17	7.468	8.373	11.268	11.378
18	7.791	8.782	11.842	11.938
19	8.110	9.188	12.413	12.494
20	8.425	9.591	12.980	13.045
21	8.736	9.993	13.545	13.593
22	9.045	10.393	14.106	14.137
23	9.350	10.791	14.665	14.678
24	9.653	11.188	15.221	15.215
25	9.952	11.583	15.774	15.748
26	10.250	11.976	16.325	16.265
27	10.545	12.369	16.873	16.779
28	10.838	12.759	17.420	17.288
29	11.128	13.149	17.964	17.796
30	11.416	13.537	18.507	18.300
31	11.703	13.924	19.047	18.802
32	11.987	14.310	19.586	

Table 5.1 - Bounds on the branching factor of the α-β pruning algorithm

6 - Conclusions and open problems

We have presented an analysis of the performance of the α-β pruning algorithm for searching a uniform tree of degree n and depth d when the values assigned to the terminal nodes are independent identically distributed random variables. The analysis takes into account both shallow and deep cut-offs and we have also considered the effect of equalities between the values assigned to the terminal nodes.

A simple formula was derived, in Section 3, to measure the number of terminal nodes examined by the α-β procedure when the bottom values are drawn from a finite range according to an arbitrary *discrete* probability distribution. Although the formula can be easily computed numerically, a direct analysis is made difficult by the presence of the probability distribution. When only two distinct values can be assigned to the terminal nodes, it is shown that the number of terminal nodes examined by the α-β procedure can be at least $O[(n/\ln n)^d]$, and, in light of the results of Section 5, this corresponds to the worst case behavior of the algorithm (over all discrete probability distributions).

A formula was then presented in the form of an integral to measure the number of terminal nodes explored by the α-β procedure when the bottom values are all distinct. An analysis of the integral shows that the branching factor of the α-β pruning algorithm is $\Theta(n/\ln n)$, a result which confirms a claim by Knuth and Moore [3] that deep cut-offs only have a second order effect on the behavior of the α-β pruning algorithm.

Although the assumption used in Sections 4 and 5 when the bottom values are all distinct is not realistic for most practical applications, the results we have derived from it give us some insight into the worst case behavior of the α-β pruning algorithm when equalities between bottom values are possible, and they are a useful complement to the formula of Section 3. Similarly, the branching factor analyzed in Section 5 provides us only with an asymptotic measure of performance for the α-β pruning algorithm (i. e., for trees of large depth). As indicated by the results of Section 3.3, however, the branching factor can also be used as a realistic measure of the worst case even for small trees.

We have measured the efficiency of the α-β pruning algorithm by the average number of terminal nodes explored

by the algorithm, it would be interesting to also obtain an estimate for the standard deviation of this number.

The scheme we have considered for assigning values to terminal nodes of a uniform tree lent itself easily to analysis; it is, however, very simplistic. Different schemes for assigning static values have been proposed in [1], [3] and [4]. Analyses of these schemes would be helpful for various applications; a step in this direction was presented in [4] for game trees of depth 2 and 3.

Acknowledgements

I wish to thank H. T. Kung and B. W. Weide for reading and commenting on earlier drafts of this paper.

References

[1] Fuller, S. H., Gaschnig, J. G., and Gillogly, J. J., Analysis of the alpha-beta pruning algorithm, Carnegie-Mellon University, Computer Science Department Report, July 1973.

[2] Gillogly, J. J., The Technology chess program, *Artificial Intelligence*, Vol. 3, No. 3, 1972, pp. 145-163.

[3] Knuth, D. E., and Moore, R. W., An analysis of alpha-beta pruning, *Artificial Intelligence*, Vol. 6, No. 4, 1975, pp. 293-326.

[4] Newborn, M. M., The efficiency of the alpha-beta search on trees with branch-dependent terminal node scores, *Artificial Intelligence*, Vol. 8, No. 2, 1977, pp. 137-153.

[5] Slagle, J. R., and Dixon, J. K., Experiments with some programs that search game trees, *Journal of the ACM*, Vol. 16, 1969, pp. 189-207.